US005794177A

United States Patent [19]

Carus et al.

[11] Patent Number: 5,794,177

[45] Date of Patent: Aug. 11, 1998

[54] **METHOD AND APPARATUS FOR MORPHOLOGICAL ANALYSIS AND GENERATION OF NATURAL LANGUAGE TEXT**

[75] Inventors: **Alwin B. Carus**, Newton; **Michael Wiesner**, West Roxbury; **Keith Boone**, Roslindale, all of Mass.

[73] Assignee: **Inso Corporation**, Boston, Mass.

[21] Appl. No.: **555,495**

[22] Filed: **Nov. 8, 1995**

Related U.S. Application Data

[63] Continuation-in-part of Ser. No. 503,981, Jul. 19, 1995, Pat. No. 5,680,628.

[51] **Int. Cl.**[6] **G06F 17/28**; G06F 17/21

[52] **U.S. Cl.** **704/9**; 704/10; 704/2; 705/5; 707/532; 707/533

[58] **Field of Search** 704/9, 10, 8, 1, 704/2, 7; 707/2, 5, 532, 533; 395/12

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,724,523	2/1988	Kucera	707/532
4,730,270	3/1988	Okajima et al.	704/2
4,771,401	9/1988	Kaufman et al.	707/533
4,862,408	8/1989	Zamora	707/102
4,864,501	9/1989	Kucera	704/8
4,864,502	9/1989	Kucera et al.	704/8
4,914,590	4/1990	Loatman et al.	704/8
4,964,044	10/1990	Kumano et al.	704/6
5,229,936	7/1993	Decker et al.	704/10
5,251,129	10/1993	Jacobs et al.	704/9
5,282,265	1/1994	Rohra Suda et al.	395/12
5,331,556	7/1994	Black, Jr. et al.	704/9
5,383,120	1/1995	Zernik	704/10
5,475,587	12/1995	Anick et al.	704/9
5,490,061	2/1996	Tolin et al.	704/2
5,559,693	9/1996	Anick et al.	704/9
5,677,835	10/1997	Carbonell et al.	395/12
5,680,628	10/1997	Carus et al.	1/1

FOREIGN PATENT DOCUMENTS

0282721A3	9/1988	European Pat. Off. .
0316743A3	5/1989	European Pat. Off. .
0327266A2	8/1989	European Pat. Off. .
0394633A2	10/1990	European Pat. Off. .
0 583 083 A2	7/1993	European Pat. Off. .
0562334A2	9/1993	European Pat. Off. .

OTHER PUBLICATIONS

Brill, Eric, "A Simple Rule-Based Part of Speech Tagger", *Third Conf. Applied Natural Lang. Processing, Proceedings of the Conference* (1992).

Schwarz, C., "Automatic Syntactic Analysis of Free Text", *J. Am. Soc. Info. Sci.* 41(6):408-417 (1990).

International Search Report mailed: Oct. 15, 1996.

Frakes, W. and Baeza-Yates, R. (eds), *Information Retrieval Data Structures and Algorithms*, PTR Prentice-Hall, Inc., ch. 7, 102-130 (1992).

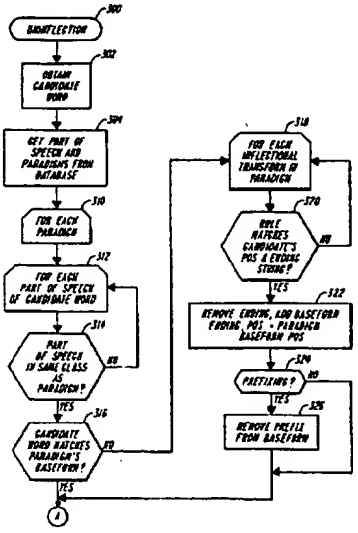
Frakes, W. and Baeza-Yates, R. (eds), *Information Retrieval Data Structures and Algorithms*, PTR Prentice-Hall, Inc., ch. 8, 131-151 (1992).

Primary Examiner—David R. Hudspeth
Assistant Examiner—Joseph Thomas
Attorney, Agent, or Firm—Lahive & Cockfield, LLP

[57] **ABSTRACT**

This invention improves information retrieval and the precision of language processing by providing an apparatus and method for organizing, utilizing, analyzing, and generating morphological data. The apparatus and method involve locating a stored lexical expression representative of a candidate word found in a stream of natural language text, identifying a paradigm for the candidate word based upon the stored lexical expression, and applying transforms contained within the identified paradigm to the candidate word.

37 Claims, 18 Drawing Sheets



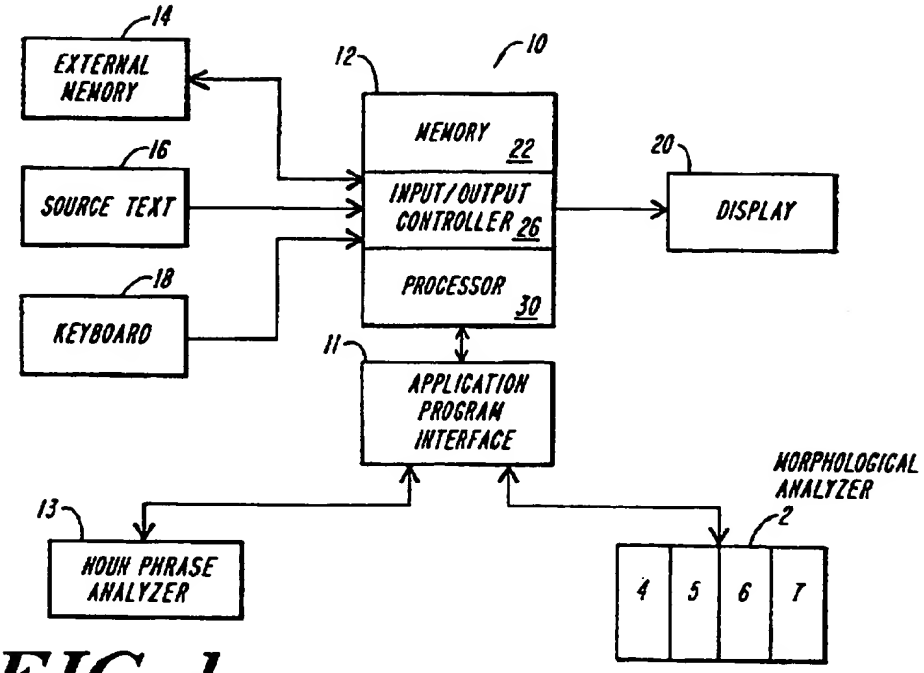

14
EXTERNAL MEMORY
12
10
MEMORY 22
16
SOURCE TEXT
INPUT/OUTPUT CONTROLLER 26
20
DISPLAY
18
KEYBOARD
PROCESSOR 30
11
APPLICATION PROGRAM INTERFACE
MORPHOLOGICAL ANALYZER
2
13
NOUN PHRASE ANALYZER
4
5
6
7

FIG. 1

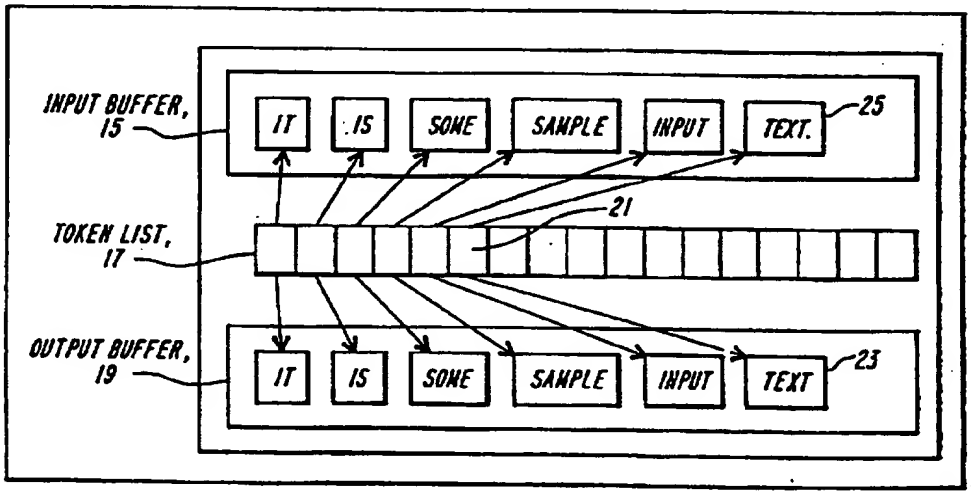

INPUT BUFFER, 15
IT
IS
SOME
SAMPLE
INPUT
TEXT.
25
TOKEN LIST, 17
21
OUTPUT BUFFER, 19
IT
IS
SOME
SAMPLE
INPUT
TEXT
23

FIG. 2

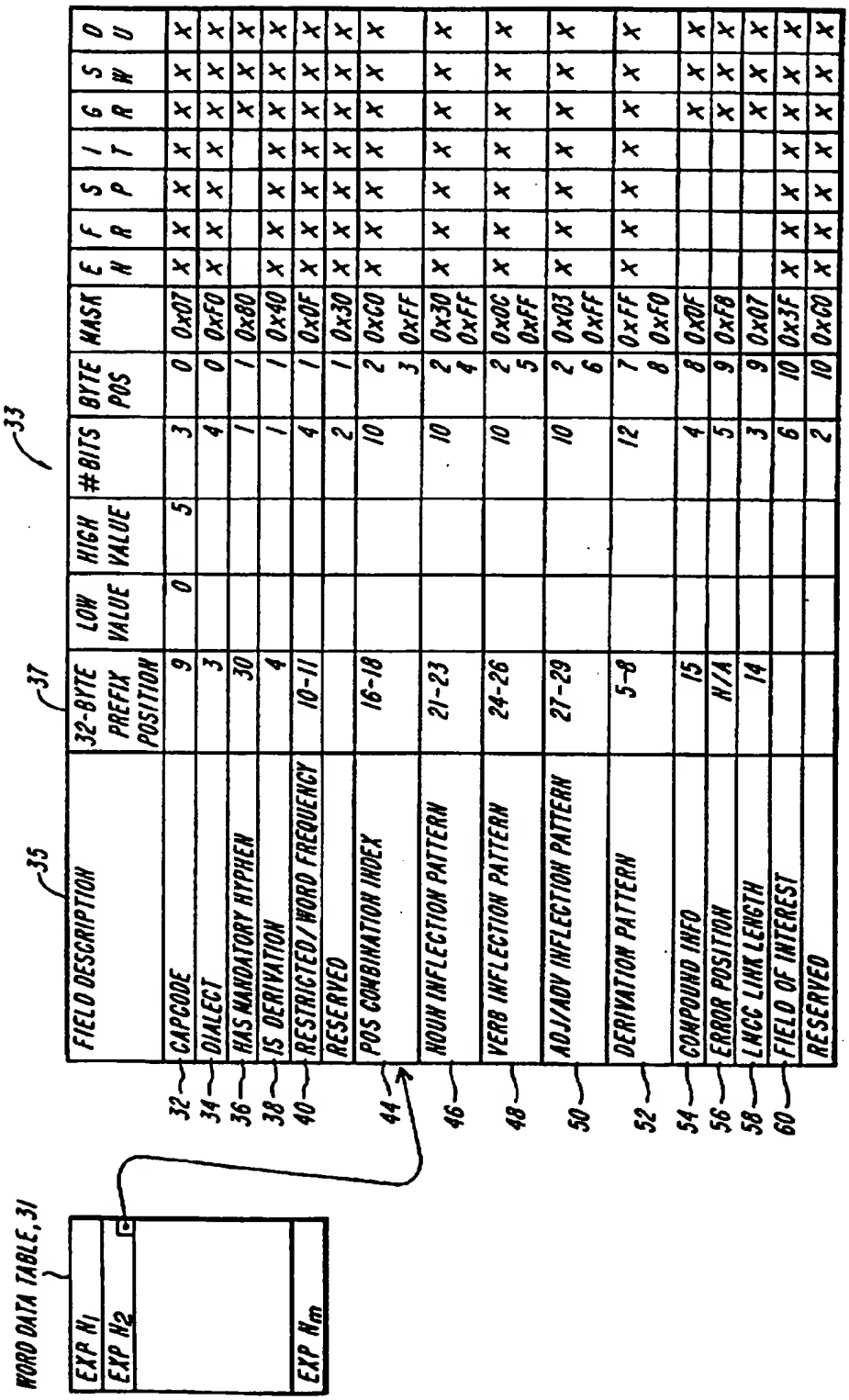

WORD DATA TABLE, 31
EXP N1
EXP N2
EXP Nm
35
37
33
32
34
36
38
40
44
46
48
50
52
54
56
58
60
FIELD DESCRIPTION
32-BYTE PREFIX POSITION
LOW VALUE
HIGH VALUE
BITS
BYTE POS
MASK
E N
F R
S P
I T
G R
S W
D U
CAPCODE 9 0 5 3 0 0x07 X X X X X X X
DIALECT 3 0 4 0 0xF0 X X X X X X X
HAS MANDATORY HYPHEN 30 1 1 0x80 X X X
IS DERIVATION 4 1 1 0x40 X X X X X X X
RESTRICTED/WORD FREQUENCY 10-11 4 1 0x0F X X X X X X X
RESERVED 2 1 0x30 X X X X X X X
POS COMBINATION INDEX 16-18 10 2 0xC0 X X X X X X X
3 0xFF
NOUN INFLECTION PATTERN 21-23 10 2 0x30 X X X X X X X
4 0xFF
VERB INFLECTION PATTERN 24-26 10 2 0x0C X X X X X X X
5 0xFF
ADJ/ADV INFLECTION PATTERN 27-29 10 2 0x03 X X X X X X X
6 0xFF
DERIVATION PATTERN 5-8 12 7 0xFF X X X X X X X
8 0xF0
COMPOUND INFO 15 4 8 0x0F X X X X
ERROR POSITION N/A 5 9 0xF8 X X X X
LNGG LINK LENGTH 14 3 9 0x07 X X X X
FIELD OF INTEREST 6 10 0x3F X X X X X X X
RESERVED 2 10 0xC0 X X X X X X X

FIG. 3

62

64 INDEX	66 POS TAG(S)	68 OEM TAG(S)
1	NN	N
2	NN$	N
3	NNS$	N
343	ABN_NN_QL_RB	N_O_R
344	ABN_NN_NNS_QL_RB	N_O_R
345	ABN	O
346	;	O
347	:	O
348	,	O

FIG. 4A

70

72 SUFFIX	74 POS INDEX
&bs	004
'am	001
...	
ôle	001

FIG. 4B

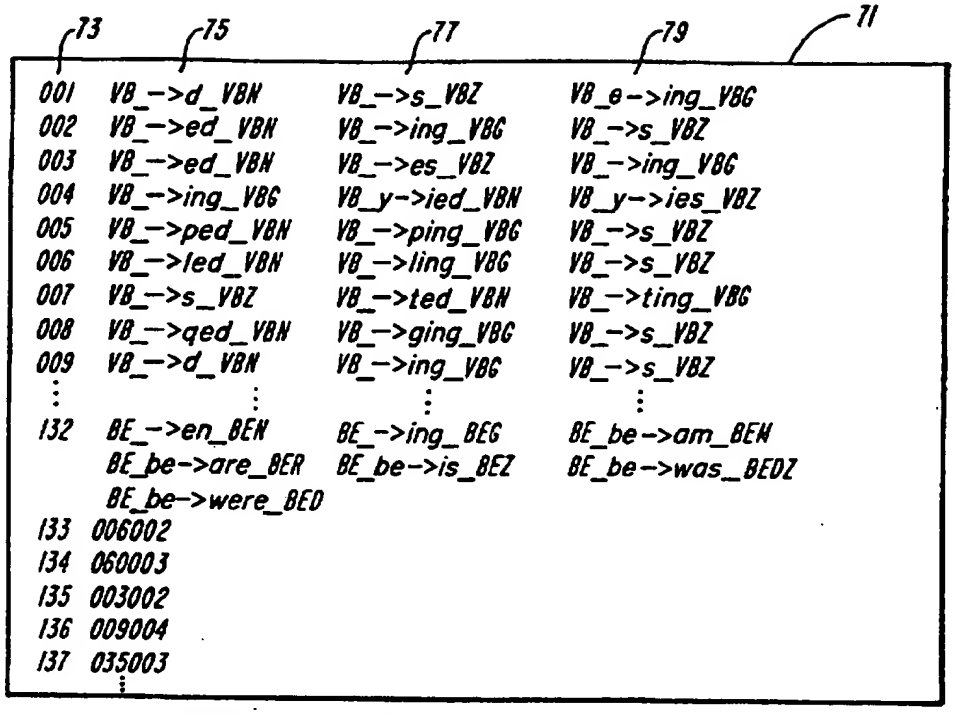

73
75
77
79
71
001 VB_->d_VBN VB_->s_VBZ VB_e->ing_VBG
002 VB_->ed_VBN VB_->ing_VBG VB_->s_VBZ
003 VB_->ed_VBN VB_->es_VBZ VB_->ing_VBG
004 VB_->ing_VBG VB_y->ied_VBN VB_y->ies_VBZ
005 VB_->ped_VBN VB_->ping_VBG VB_->s_VBZ
006 VB_->led_VBN VB_->ling_VBG VB_->s_VBZ
007 VB_->s_VBZ VB_->ted_VBN VB_->ting_VBG
008 VB_->qed_VBN VB_->ging_VBG VB_->s_VBZ
009 VB_->d_VBN VB_->ing_VBG VB_->s_VBZ
132 BE_->en_BEN BE_->ing_BEG BE_be->am_BEM
BE_be->are_BER BE_be->is_BEZ BE_be->was_BEDZ
BE_be->were_BED
133 006002
134 060003
135 003002
136 009004
137 035003

FIG. 4C

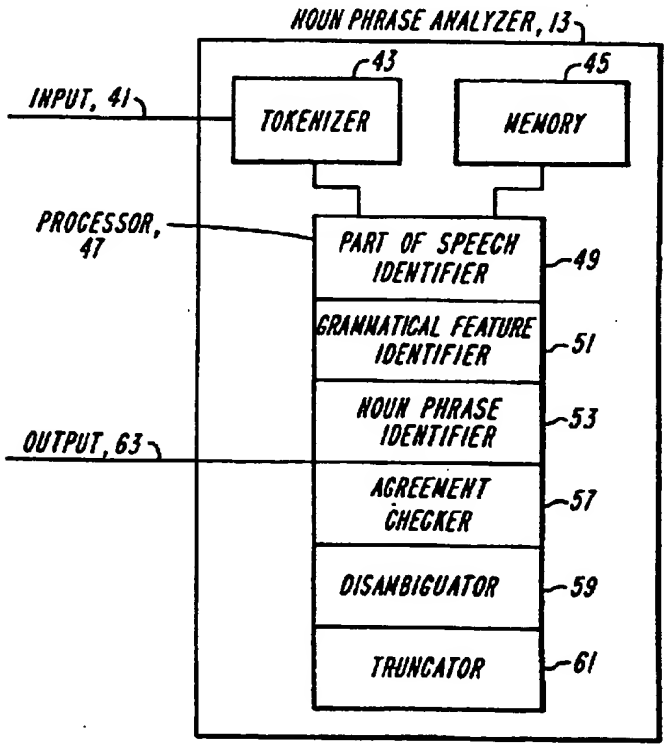

NOUN PHRASE ANALYZER, 13
43
45
INPUT, 41
TOKENIZER
MEMORY
PROCESSOR, 47
PART OF SPEECH IDENTIFIER
49
GRAMMATICAL FEATURE IDENTIFIER
51
NOUN PHRASE IDENTIFIER
53
OUTPUT, 63
AGREEMENT CHECKER
57
DISAMBIGUATOR
59
TRUNCATOR
61

FIG. 6

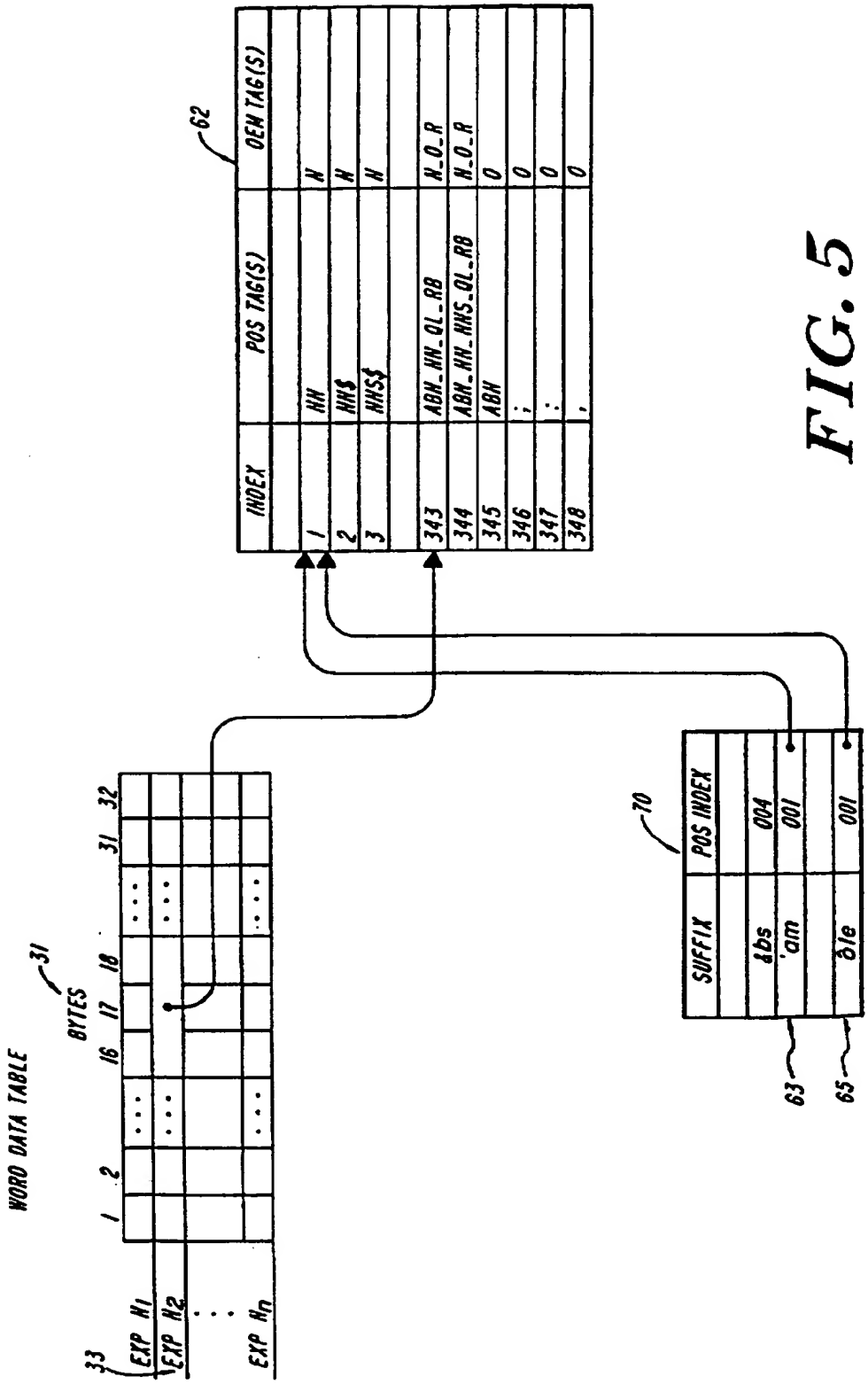

62
INDEX
POS TAG(S)
OEN TAG(S)
1
HW
N
2
HWS
N
3
HWSS
N
343
ABH_HW_QL_RB
N_O_R
344
ABH_HW_HWS_QL_RB
N_O_R
345
ABH
0
346
;
0
347
:
0
348
,
0
70
SUFFIX
POS INDEX
004
001
001
63
65
31
BYTES
WORD DATA TABLE
1
2
16
17
18
31
32
33
EXP H1
EXP H2
EXP Hn

FIG. 5

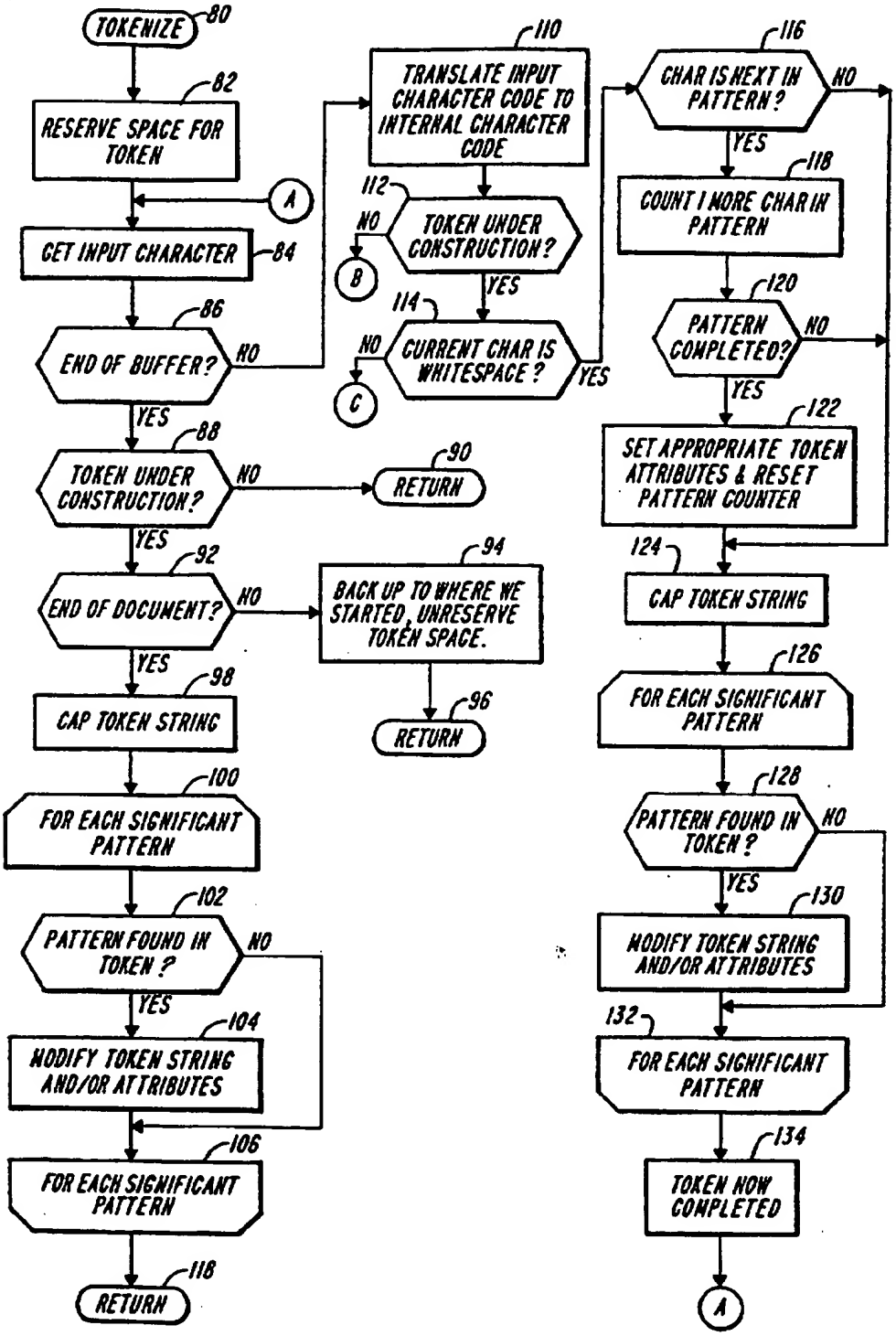
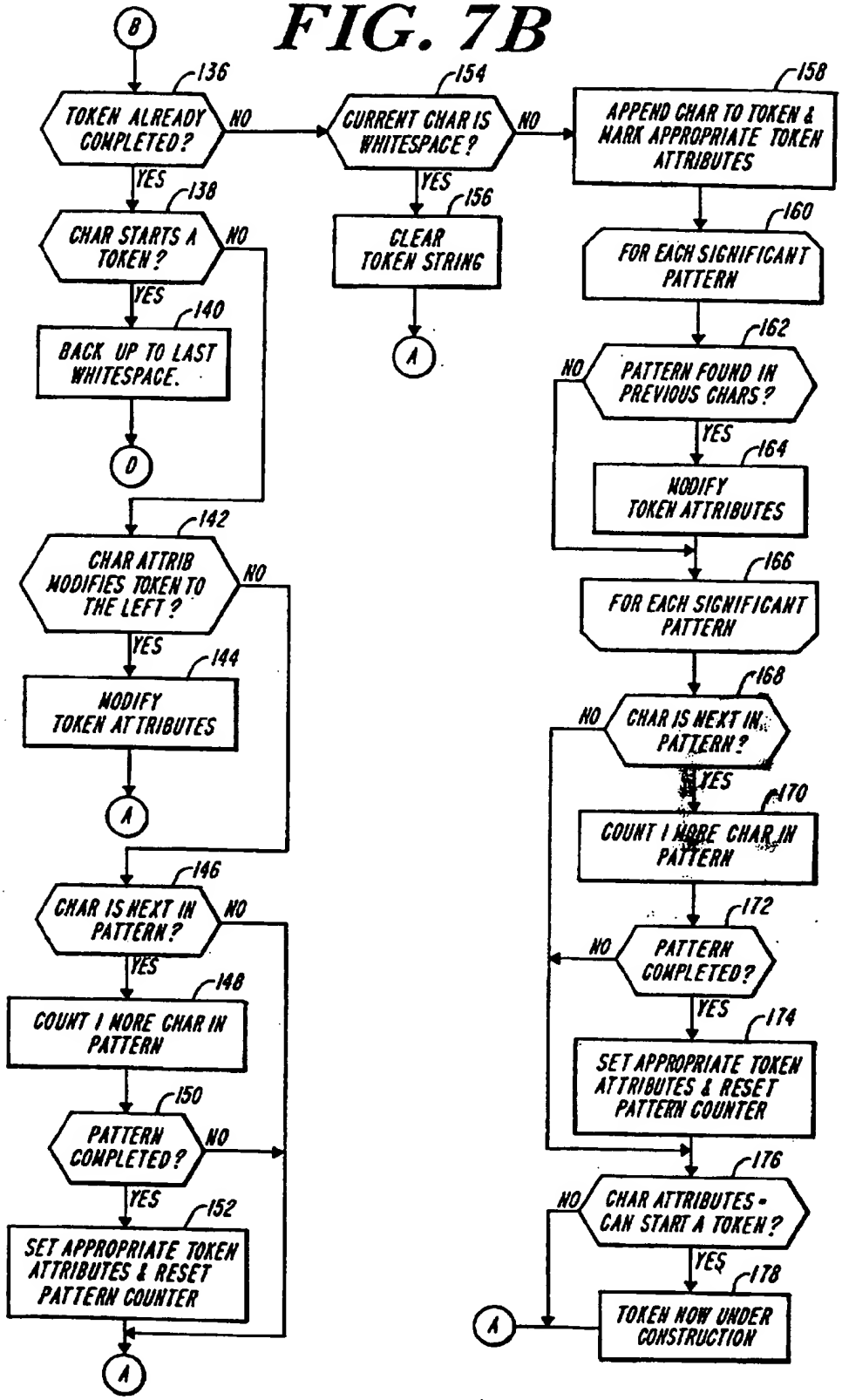

TOKENIZE
80
RESERVE SPACE FOR TOKEN
82
A
GET INPUT CHARACTER
84
END OF BUFFER?
86
NO
YES
TOKEN UNDER CONSTRUCTION?
88
NO
RETURN
90
YES
END OF DOCUMENT?
92
NO
BACK UP TO WHERE WE STARTED, UNRESERVE TOKEN SPACE.
94
YES
RETURN
96
CAP TOKEN STRING
98
FOR EACH SIGNIFICANT PATTERN
100
PATTERN FOUND IN TOKEN ?
102
NO
YES
MODIFY TOKEN STRING AND/OR ATTRIBUTES
104
FOR EACH SIGNIFICANT PATTERN
106
RETURN
118
TRANSLATE INPUT CHARACTER CODE TO INTERNAL CHARACTER CODE
110
TOKEN UNDER CONSTRUCTION?
112
NO
B
YES
CURRENT CHAR IS WHITESPACE ?
114
NO
C
YES
CHAR IS NEXT IN PATTERN?
116
NO
YES
COUNT I MORE CHAR IN PATTERN
118
PATTERN COMPLETED?
120
NO
YES
SET APPROPRIATE TOKEN ATTRIBUTES & RESET PATTERN COUNTER
122
CAP TOKEN STRING
124
FOR EACH SIGNIFICANT PATTERN
126
PATTERN FOUND IN TOKEN ?
128
NO
YES
MODIFY TOKEN STRING AND/OR ATTRIBUTES
130
FOR EACH SIGNIFICANT PATTERN
132
TOKEN NOW COMPLETED
134
A

FIG. 7A

FIG. 7B


B
136
TOKEN ALREADY COMPLETED?
NO
YES
138
CHAR STARTS A TOKEN?
NO
YES
140
BACK UP TO LAST WHITESPACE.
D
142
CHAR ATTRIB MODIFIES TOKEN TO THE LEFT?
NO
YES
144
MODIFY TOKEN ATTRIBUTES
A
146
CHAR IS NEXT IN PATTERN?
NO
YES
148
COUNT 1 MORE CHAR IN PATTERN
150
PATTERN COMPLETED?
NO
YES
152
SET APPROPRIATE TOKEN ATTRIBUTES & RESET PATTERN COUNTER
A
154
CURRENT CHAR IS WHITESPACE?
NO
YES
156
CLEAR TOKEN STRING
A
158
APPEND CHAR TO TOKEN & MARK APPROPRIATE TOKEN ATTRIBUTES
160
FOR EACH SIGNIFICANT PATTERN
162
NO
PATTERN FOUND IN PREVIOUS CHARS?
YES
164
MODIFY TOKEN ATTRIBUTES
166
FOR EACH SIGNIFICANT PATTERN
168
NO
CHAR IS NEXT IN PATTERN?
YES
170
COUNT 1 MORE CHAR IN PATTERN
172
NO
PATTERN COMPLETED?
YES
174
SET APPROPRIATE TOKEN ATTRIBUTES & RESET PATTERN COUNTER
176
NO
CHAR ATTRIBUTES - CAN START A TOKEN?
YES
178
TOKEN NOW UNDER CONSTRUCTION
A

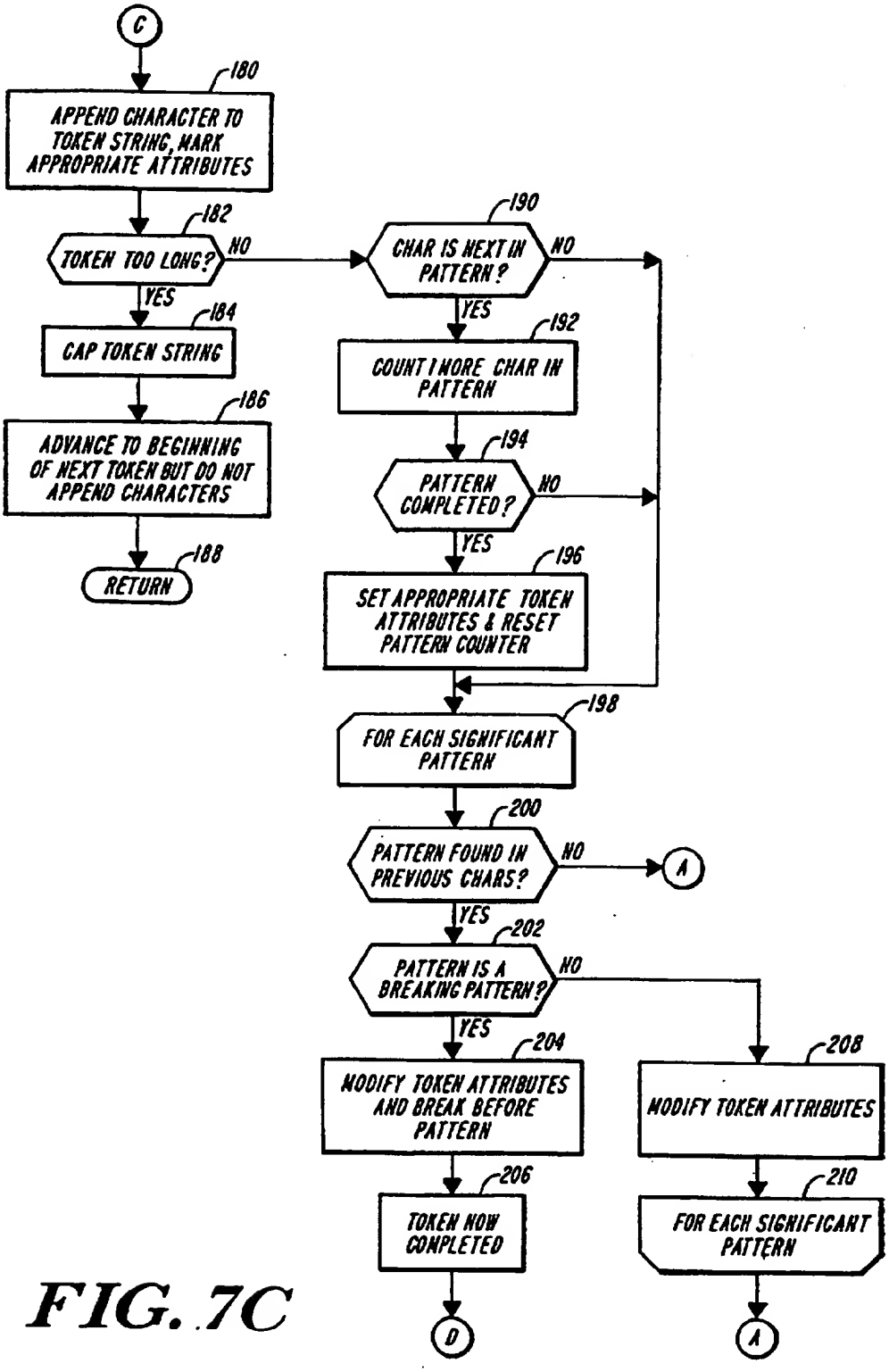

C
180
APPEND CHARACTER TO TOKEN STRING, MARK APPROPRIATE ATTRIBUTES
182
TOKEN TOO LONG?
NO
YES
184
CAP TOKEN STRING
186
ADVANCE TO BEGINNING OF NEXT TOKEN BUT DO NOT APPEND CHARACTERS
188
RETURN
190
CHAR IS NEXT IN PATTERN?
NO
YES
192
COUNT I MORE CHAR IN PATTERN
194
PATTERN COMPLETED?
NO
YES
196
SET APPROPRIATE TOKEN ATTRIBUTES & RESET PATTERN COUNTER
198
FOR EACH SIGNIFICANT PATTERN
200
PATTERN FOUND IN PREVIOUS CHARS?
NO
A
YES
202
PATTERN IS A BREAKING PATTERN?
NO
YES
204
MODIFY TOKEN ATTRIBUTES AND BREAK BEFORE PATTERN
206
TOKEN NOW COMPLETED
D
208
MODIFY TOKEN ATTRIBUTES
210
FOR EACH SIGNIFICANT PATTERN
A

FIG. 7C

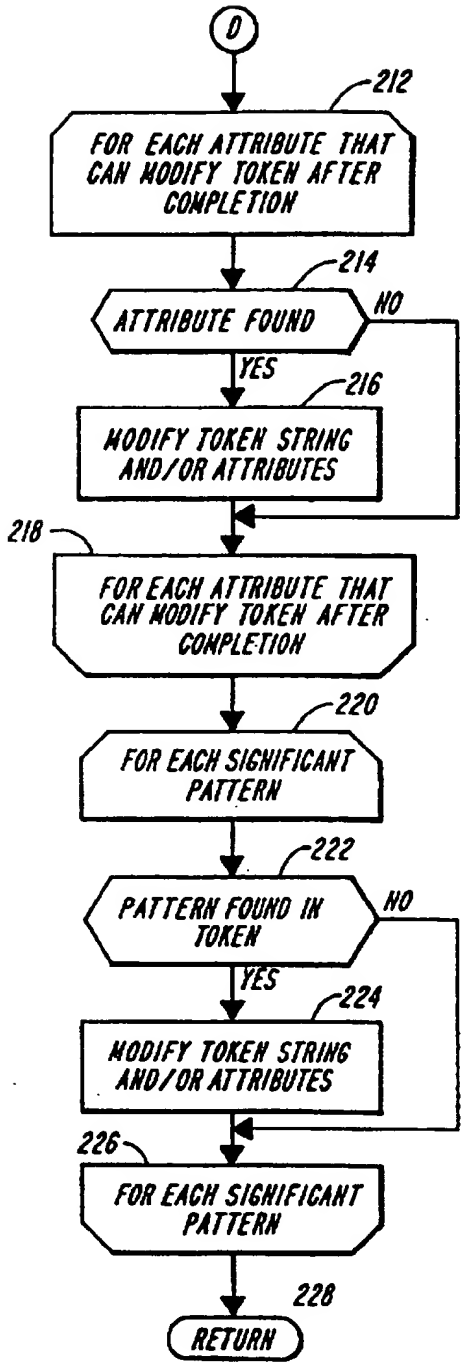

0
212
FOR EACH ATTRIBUTE THAT CAN MODIFY TOKEN AFTER COMPLETION
214
ATTRIBUTE FOUND
NO
YES
216
MODIFY TOKEN STRING AND/OR ATTRIBUTES
218
FOR EACH ATTRIBUTE THAT CAN MODIFY TOKEN AFTER COMPLETION
220
FOR EACH SIGNIFICANT PATTERN
222
PATTERN FOUND IN TOKEN
NO
YES
224
MODIFY TOKEN STRING AND/OR ATTRIBUTES
226
FOR EACH SIGNIFICANT PATTERN
228
RETURN

FIG. 7D

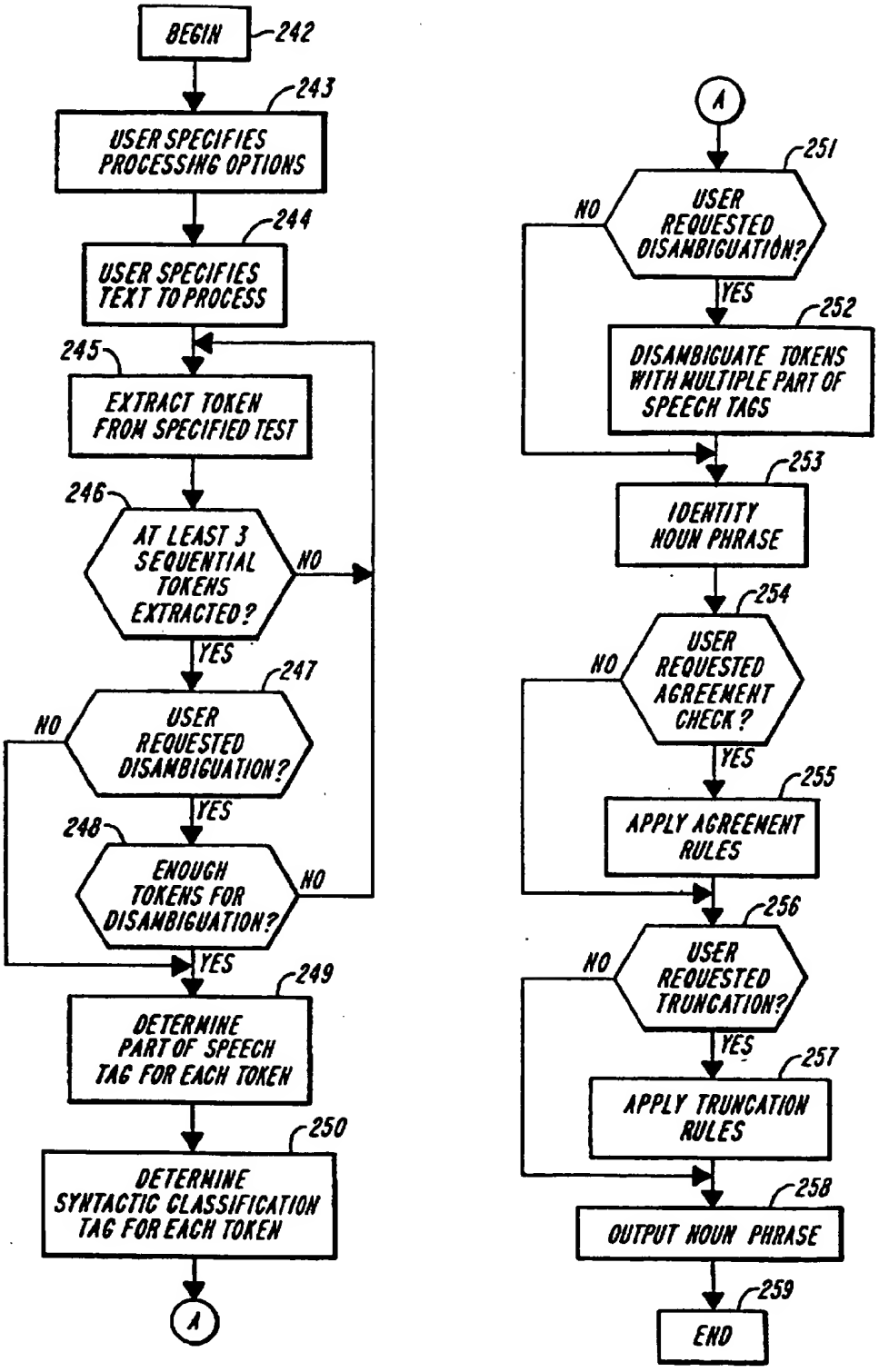

BEGIN
242
USER SPECIFIES PROCESSING OPTIONS
243
USER SPECIFIES TEXT TO PROCESS
244
EXTRACT TOKEN FROM SPECIFIED TEST
245
AT LEAST 3 SEQUENTIAL TOKENS EXTRACTED?
246
NO
YES
USER REQUESTED DISAMBIGUATION?
247
NO
YES
ENOUGH TOKENS FOR DISAMBIGUATION?
248
NO
YES
DETERMINE PART OF SPEECH TAG FOR EACH TOKEN
249
DETERMINE SYNTACTIC CLASSIFICATION TAG FOR EACH TOKEN
250
A
A
USER REQUESTED DISAMBIGUATION?
251
NO
YES
DISAMBIGUATE TOKENS WITH MULTIPLE PART OF SPEECH TAGS
252
IDENTITY NOUN PHRASE
253
USER REQUESTED AGREEMENT CHECK?
254
NO
YES
APPLY AGREEMENT RULES
255
USER REQUESTED TRUNCATION?
256
NO
YES
APPLY TRUNCATION RULES
257
OUTPUT NOUN PHRASE
258
END
259

FIG. 8

261 RULE	260 (i-2)	262 (i-1)	264 (i)	266 (i+1)
1			IF BEGINNING OF SENTENCE & CAPCODE>000 & PART-OF-SPEECH TAG = NOUN, THEN COERCE PRIMARY PART-OF-SPEECH TAG TO SINGULAR COMMON NOUN 268	
2	PRIMARY PART-OF-SPEECH TAG=ARTICLE 270		IF PRIMARY PART-OF-SPEECH TAG = VERB OR SECOND POSSESSIVE PRONOUN OR EXCLAMATION OR VERB PAST TENSE FORM & SECONDARY PART-OF-SPEECH TAG = SINGULAR COMMON NOUN, THEN PROMOTE SECONDARY PART-OF-SPEECH TAG 272	
3		PART-OF-SPEECH TAG = VERB INFINITIVE OR SINGULAR COMMON NOUN 274	IF PRIMARY PART-OF-SPEECH TAG = VERB OR SECOND POSSESSIVE PRONOUN OR EXCLAIMATION OR VERB PAST TENSE FORM & SECONDARY PART-OF-SPEECH TAG = SINGULAR COMMON NOUN, THEN PROMOTE SECONDARY PART-OF-SPEECH TAG 276	
4		PART-OF-SPEECH TAG = MODAL AUXILLARY OR SINGULAR COMMON NOUN 278	IF PRIMARY PART-OF-SPEECH TAG = MODAL AUXILLIARY & SECONDARY PART-OF-SPEECH TAG = SINGULAR COMMON NOUN, THEN PROMOTE SECONDARY PART-OF-SPEECH TAG 280	PART-OF-SPEECH TAG = INFINITIVE 282
5		PART-OF-SPEECH TAG = VERB INFINITIVE OR SINGULAR COMMON NOUN 284	IF PRIMARY PART-OF-SPEECH TAG = VERB & SECONDARY PART-OF-SPEECH TAG = ADJECTIVE, THEN PROMOTE SECONDARY PART-OF-SPEECH TAG 286	
6		PART-OF-SPEECH TAG = VERB INFINITIVE OR SINGULAR COMMON NOUN 287	IF PRIMARY PART-OF-SPEECH TAG = VERB & SECONDARY PART-OF-SPEECH TAG = COMPARATIVE ADJECTIVE, THEN PROMOTE SECONDARY PART-OF-SPEECH TAG 288	

FIG. 9

```
IF AGREEMENT CHECKS TWO TOKENS T1 AND T2:

  REDUCE POS1...POSn ON T1 TO NOUN PHRASE TAGS (CN, JJ*, NM, NN*, ON)
  IF ANY OF THE REMAINING TAGS IS MARKED AS 'MATCHED'
    REDUCE THE SET TO ONLY 'MATCHED' TAGS

  REDUCE POS1...POSn ON T2 TO AGREEMENT TAGS (CN, JJ*, NM, NN*, ON)
  IF ANY OF THE REMAINING TAGS IS MARKED AS 'MATCHED'
    REDUCE THE SET TO ONLY 'MATCHED' TAGS

  FOR EVERY POSi ON T1
      FOR EVERY POSj ON T2
          IF EITHER POSi OR POSj IS A CN, NM, OR ON
             MARK POSi AND POSj  AS 'MATCHED'
          ELSE IF LANGUAGE IS FR/IT/SP:
                  IF THE INTERSECTION OF NUMBER ON POSi AND POSj IS NOT EMPTY
AND
                     THE INTERSECTION OF GENDER ON POSi AND POSj IS NOT
EMPTY,
                     MARK POSi AND POSj AS 'MATCHED'
                ELSE IF LANGUAGE IS GR:
                     IF THE INTERSECTION OF NUMBER ON POSi AND POSj IS NOT
EMPTY AND
                        THE INTERSECTION OF GENDER ON POSi AND POSj IS NOT
EMPTY,
                        THE INTERSECTION OF CASE ON POS  AND POS  IS NOT
EMPTY,
                        MARK POSi AND POSj  AS 'MATCHED'

  IF AT LEAST ONE POSi ON T2 IS MARKED AS MATCHED, T1 AND T2 AGREE
  ELSE T1 AND T2 DON'T AGREE.
```

FIG. 10

```
IF TRUNCATION SWITCH IS TURNED ON:

  IF NP CONSISTS OF 2 ELEMENTS ONLY:
    RETURN IT

  ELSE IF LANGUAGE IS EN/GR:
        RETURN LAST TWO ELEMENTS OF NP                    (1)

      ELSE IF LANGUAGE IS FR/IT/SP:
            IF NP CONTAINS SEQUENCE 'NN*+ DE + NN*':
              RETURN 'NN*+ DE + NN*'                      (2)

            ELSE

                FIND THE FIRST TOKEN IN THE NP WHICH
                HAS POS NN*

                 IF THIS NN* IS FOLLOWED BY ANOTHER TOKEN:
                    RETURN THE NN* PLUS THE FOLLOWING TOKEN  (3)

                 ELSE RETURN THE NN* PLUS THE PRECEDING
                      TOKEN                               (4)
```

FIG. 11

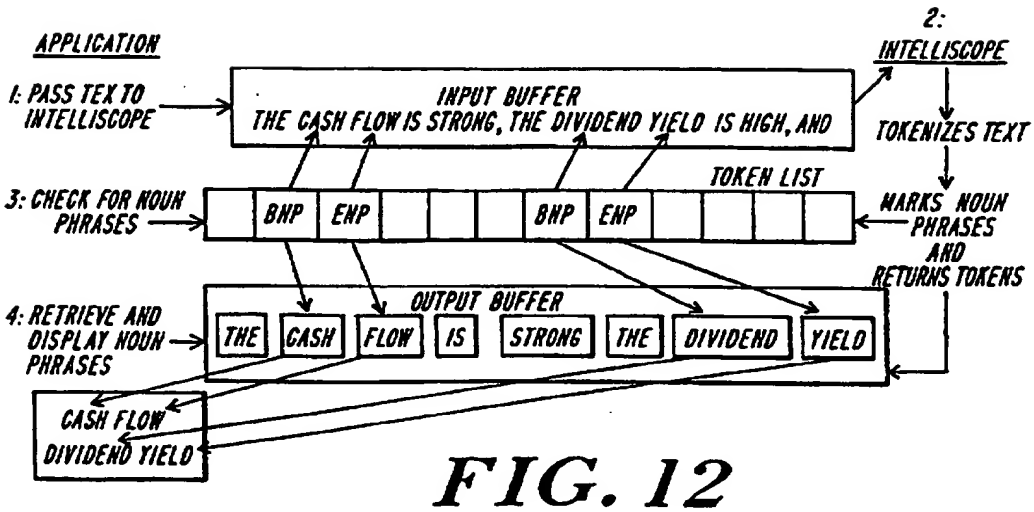


FIG. 12

```
1  FOR EACH GRAMMATICAL FIELD TYPE SELECTED FROM NOUN, VERB, OR ADVERB/ADJECTIVE
2    IF THERE ISN'T A MORPHOLOGICAL PARADIGM OF THAT TYPE FOR THE WORD
3      CONTINUE
4    END
5    IF THE RULE IS A PORTMANTEAU RULE
6      LET LIST BE THE LIST OF MORPHOLOGICAL PARADIGMS FROM THE PORTMANTEAU RULE
7    ELSE
8      LET LIST BE THIS MORPHOLOGICAL PARADIGM
9    END IF
10   FOR EACH MORPHOLOGICAL PARADIGM IN LIST
11     FOR EACH POS TAG IN THE POS COMBO ENTRY
12       & IF THE POS TAG IS NOT FOUND WITHIN THE MORPHOLOGICAL PARADIGM FOR THIS GRAMMATICAL FIELD
13       & THEN
14         & CONTINUE
15       & END IF
16       IF THE POS TAG MATCHES THE BASE POS, THEN
17         MARK THE WORD AS A BASEFORM
18         SET THE POS BIT ACCORDING TO THE GRAMMATICAL FIELD TYPE
19         IF DERIVING
20           CALL DERIVATION MODULE
21         END IF
22         IF INFLECTING
23           CALL INFLECTION MODULE WITH THIS PARADIGM
24         END IF
25         CONTINUE
26       END IF
27       FOR EACH MORPHOLOGICAL TRANSFORM IN THE PARADIGM
28         IF THE POS TAG MATCHES A MORPHOLOGICAL TRANSFORM POS TAG AND THE MORPHOLOGICAL PATTERN
29           OF THE MORPHOLOGICAL TRANSFORM MATCHES A CHARACTER STRING IN THE CANDIDATE WORD
30         THEN
31           APPLY THE MORPHOLOGICAL TRANSFORM TO PRODUCE THE BASEFORM
32           IF THE MORPHOLOGICAL TRANSFORM HAS THE PREFIX FLAG SET
33               LOOK UP THE PREFIX IN THE INFLECTION PREFIX TABLE
34               APPLY THE PREFIX TRANSFORMATION TO THE WORD
35           END
36           SET THE POS BIT ACCORDING TO THE PATTERN TYPE
37           IF THE BASEFORM IS A DUPLICATE
38               REMOVE IT
39           ELSE
40           IF THE INFLECTION DOESN'T VERIFY
41               REMOVE IT
42           ELSE
43               IF DERIVING
44                 CALL THE DERIVATION MODULE
45               END IF
46               IF INFLECTING
47                 CALL THE INFLECTION MODULE
48               END IF
49             END IF
50           END IF
51         END FOR
52       END FOR
53     END FOR
54   END FOR
```

FIG. 13

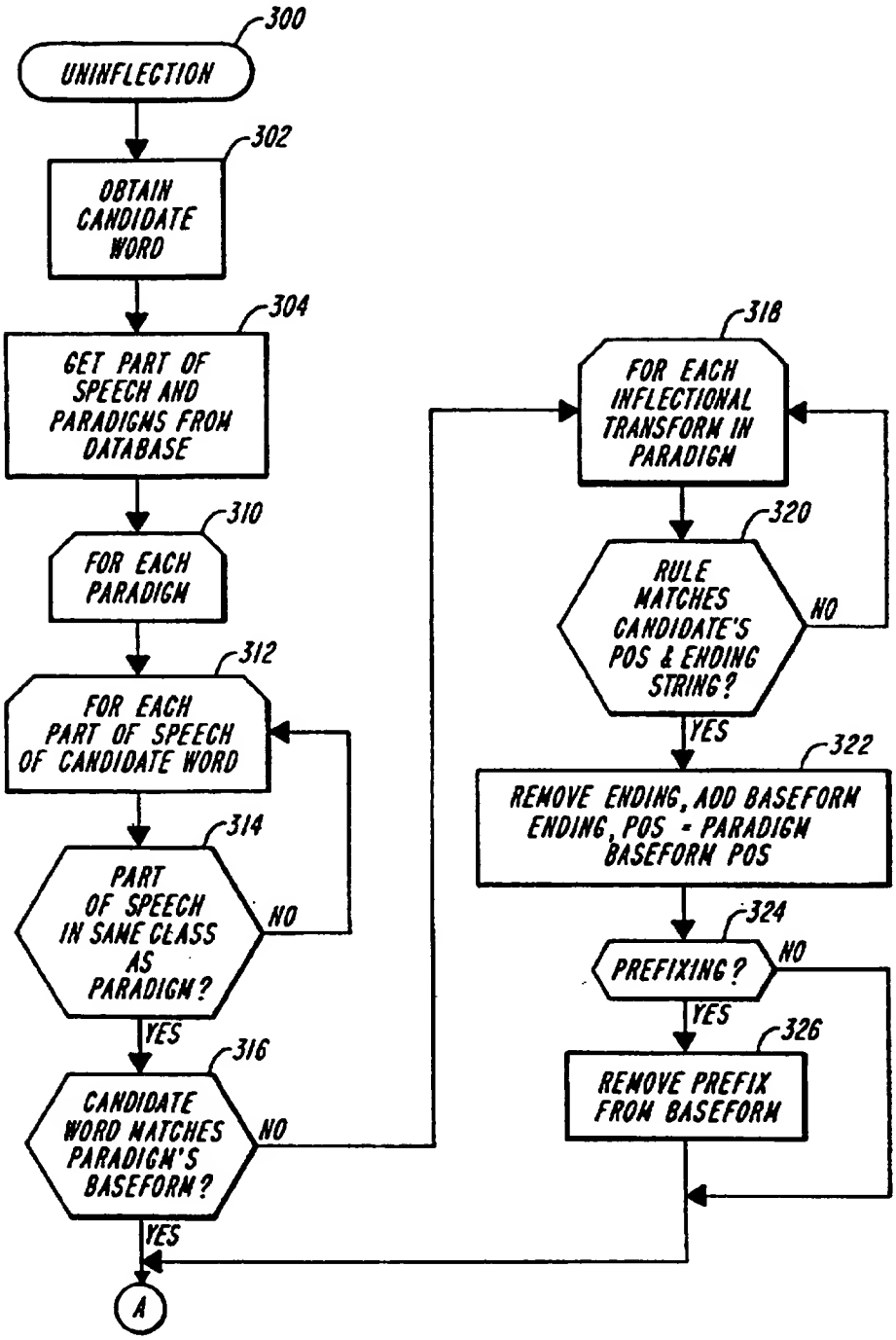

300
UNINFLECTION
302
OBTAIN CANDIDATE WORD
304
GET PART OF SPEECH AND PARADIGMS FROM DATABASE
310
FOR EACH PARADIGM
312
FOR EACH PART OF SPEECH OF CANDIDATE WORD
314
PART OF SPEECH IN SAME CLASS AS PARADIGM?
NO
YES
316
CANDIDATE WORD MATCHES PARADIGM'S BASEFORM?
NO
YES
A
318
FOR EACH INFLECTIONAL TRANSFORM IN PARADIGM
320
RULE MATCHES CANDIDATE'S POS & ENDING STRING?
NO
YES
322
REMOVE ENDING, ADD BASEFORM ENDING, POS - PARADIGM BASEFORM POS
324
PREFIXING?
NO
YES
326
REMOVE PREFIX FROM BASEFORM

FIG. 14

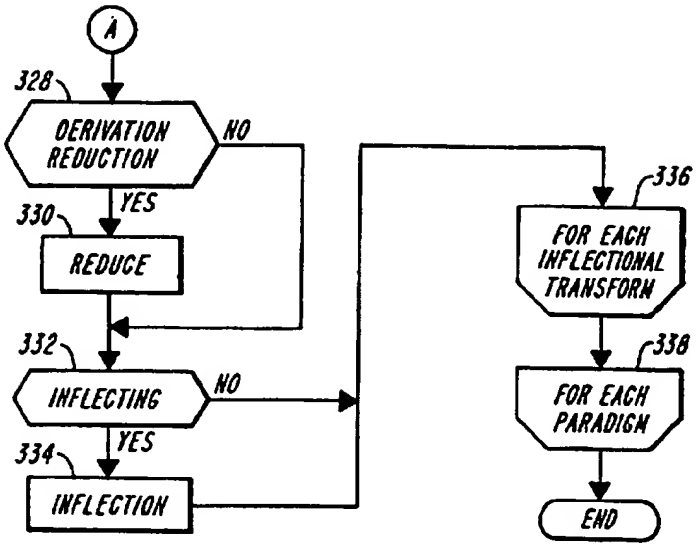


FIG. 14
(CONTINUED)

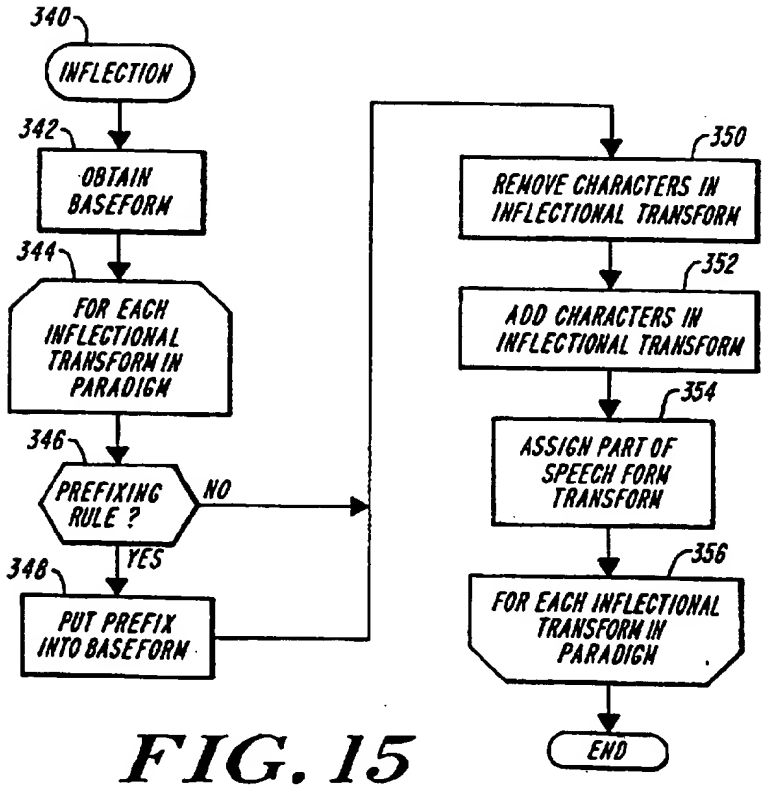


FIG. 15

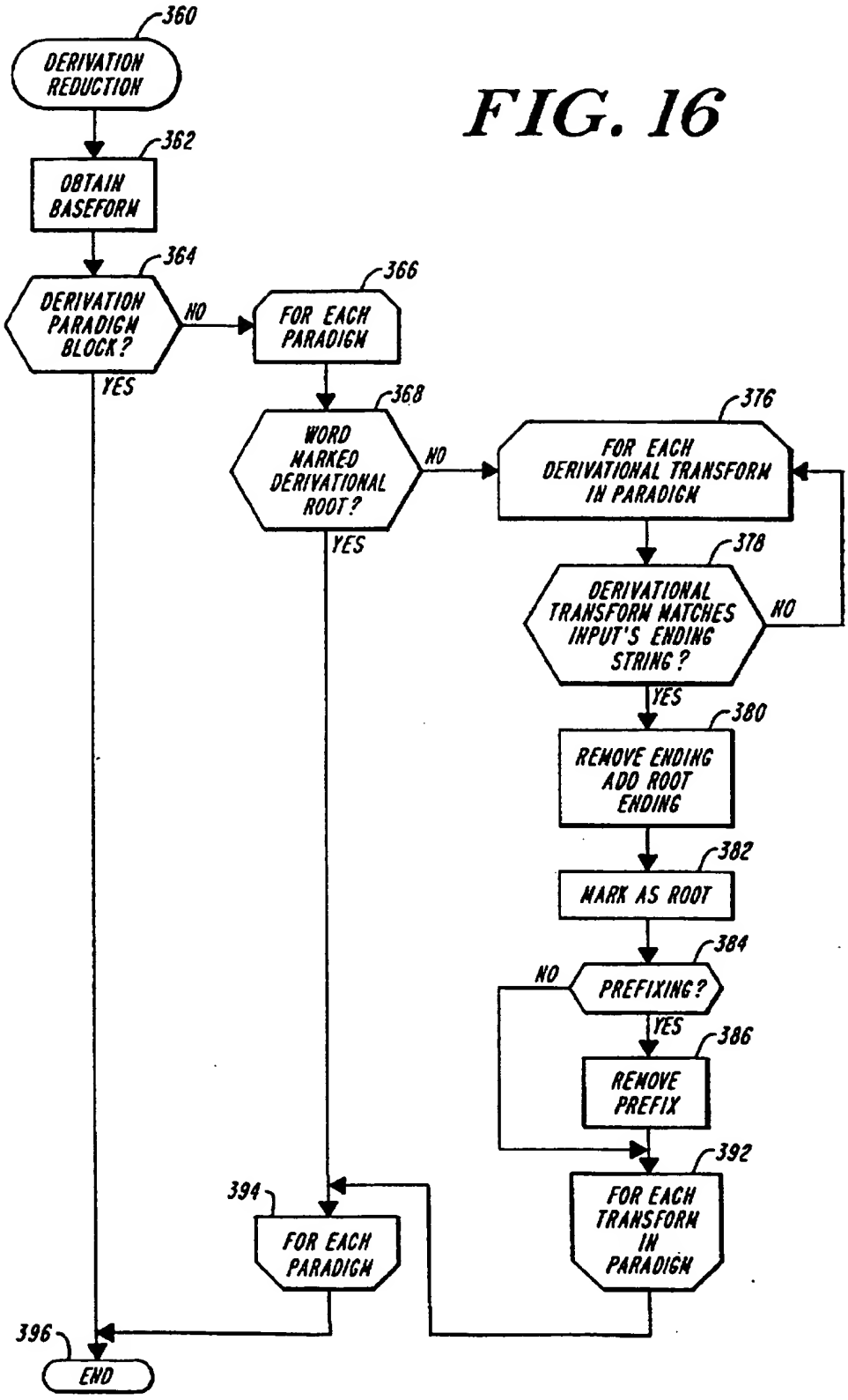

360
DERIVATION REDUCTION
362
OBTAIN BASEFORM
364
DERIVATION PARADIGM BLOCK?
NO
YES
366
FOR EACH PARADIGM
368
WORD MARKED DERIVATIONAL ROOT?
NO
YES
376
FOR EACH DERIVATIONAL TRANSFORM IN PARADIGM
378
DERIVATIONAL TRANSFORM MATCHES INPUT'S ENDING STRING?
NO
YES
380
REMOVE ENDING ADD ROOT ENDING
382
MARK AS ROOT
384
NO
PREFIXING?
YES
386
REMOVE PREFIX
392
FOR EACH TRANSFORM IN PARADIGM
394
FOR EACH PARADIGM
396
END

FIG. 16

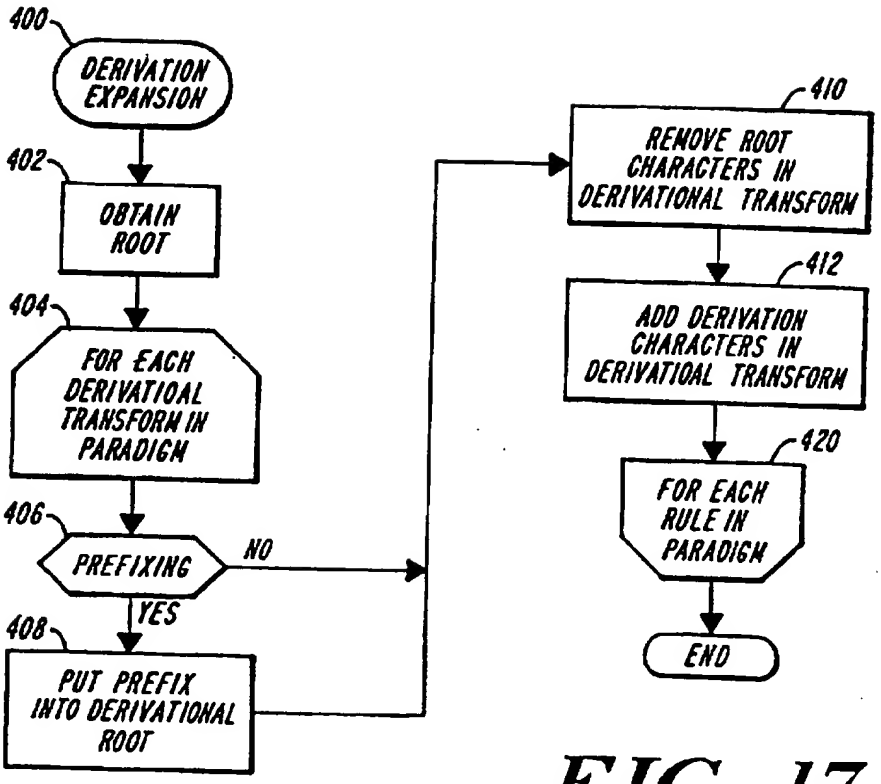

400
DERIVATION EXPANSION
402
OBTAIN ROOT
404
FOR EACH DERIVATIONAL TRANSFORM IN PARADIGM
406
PREFIXING
NO
YES
408
PUT PREFIX INTO DERIVATIONAL ROOT
410
REMOVE ROOT CHARACTERS IN DERIVATIONAL TRANSFORM
412
ADD DERIVATION CHARACTERS IN DERIVATIONAL TRANSFORM
420
FOR EACH RULE IN PARADIGM
END

FIG. 17

METHOD AND APPARATUS FOR MORPHOLOGICAL ANALYSIS AND GENERATION OF NATURAL LANGUAGE TEXT

RELATED APPLICATION

This application is a continuation-in-part of the commonly-owned U.S. patent application Ser. No. 08/503,981, filed on Jul. 19, 1995, U.S. Pat. No. 5,680,628.

BACKGROUND OF THE INVENTION

This invention relates to methods and apparatus of automated language analysis systems. It provides systems embodied in a computer for receiving digitally encoded text composed in a natural language, and it provides systems for the grammatical analysis and generation of encoded text. More particularly, the invention relates to a system for incorporating morphological analysis and generation of natural language text.

Lexical morphology involves the study and description of word formation in a language, and in particular emphasizes the examination of inflections, derivations, and compound and cliticized words. Inflectional morphology refers to the study of the alternations of the form of words by adding affixes or by changing the form of a base in order to indicate grammatical features such as number, gender, case, person, mood, or voice (e.g., the inflected forms of book: book, book's, books, and books'). Derivational morphology refers to the study of the processes by which words are formed from existing words or bases by adding or removing affixes (e.g., singer from sing) or by changing the shape of the word or base (e.g., song from sing). Compounding refers to the process by which words are formed from two or more elements which are themselves words or special combining forms of words (e.g., the German Versicherungsgesellschaft [insurance company] consisting of Versicherung+s+ Gesellschaft). Cliticizing refers to the process by which special words or particles which have no independent accent are combined with stressed content words (e.g., the French l'cole consists of the preposed enclitic le [the] and the word cole [school]).

Lexical morphology provides information useful in the implementation of natural language data processing systems. For instance, data processing systems using knowledge of lexical morphology can enhance indexing and searching techniques.

Current natural language text processing systems known in the art attempt to effectively use natural language text by processing the natural language text with computer systems. These computer systems typically include software modules for representing the natural language text and processing modules for manipulating the natural language text. The software modules further include software elements, alternatively called "lexicon modules", "dictionaries", or "databases", that contain words and related linguistic knowledge for understanding and representing natural language text. The processing module can contain a plurality of analyzing modules. In operation, the processing modules operate upon the natural language text in association with the software modules to generate a computer representation of the natural language text and to effectively use the text.

Many text processing systems utilize crude affix stripping methods called "stemmers" to analyze natural language text. Other more sophisticated, linguistically based morphological systems reduce all word forms to the same constant length character string, which is itself not necessarily a word. This "stem" portion of the word remains invariant during the morphological analysis. For example, a sophisticated morphological system might strip off the varying suffix letters to map every word to the longest common prefix character string. Thus, all the forms of arrive (i.e., arrive, arrives, arrived, and arriving) are stripped back to the longest common character string, arriv (without an e). Note that this procedure does not map forms of arrive back to arrive because the e character fails to appear in arriving. These same algorithms convert all inflected forms of swim to sw because this is the longest common substring. Both stemming and more refined morphological analysis systems, however, have proven difficult to implement because of the special mechanisms required to deal with irregular morphological patterns.

Often an exception dictionary is provided to deal with irregularities in inflection and derivation, but as a result of the number of entries in this exception dictionary it can become large and cumbersome. One alternative to using a large exception dictionary involves forming a system having a smaller, yet incomplete, exception dictionary. Although this alternative is not as cumbersome, the incomplete data structure rapidly forms inaccurate representations of the natural language text under consideration. These two alternative lexicons exemplify the problems involved in prior art systems, i.e., the difficulty in using the lexicons and the inaccuracies within the lexicon. Accordingly, many in the field have concluded that current stemming procedures cannot significantly improve coverage of the stemming algorithm without reducing their accuracy.

Another drawback of these prior systems is their inability to generate all the variant forms from a given stem. Traditional stemming algorithms can be used for finding stems, but not for generating inflections or derivations. Furthermore, these techniques are not linguistically general and require different algorithms and particular exception dictionaries for each natural language.

Accordingly, it is an object of the invention to provide an improved computer system for analyzing natural language text and for improving the precision and recall of information retrieval systems. Another object of the invention is to provide a morphological analysis and generation system that identifies the inflectional and derivational characteristics of a candidate word in the natural language text.

Another object of the invention is to provide a system having a strip and add processor that accurately constructs variable-length inflectional and derivational bases of words contained in natural language text.

Additional objects of the invention include providing a morphological analysis system capable of identifying morphological paradigms associated with candidate words contained in a natural language text in a manner that improves efficiency, increases recall, and increases the precision of index pre-processing, search pre-processing, and search expansion techniques.

SUMMARY OF THE INVENTION

The invention attains these and other objects by providing a unique system for organizing, utilizing, and analyzing morphological data associated with a candidate word obtained from a stream of natural language text. The invention includes a processor for analyzing the stream of text and for manipulating digital signals representative of morphological pattern, and a memory element for storing digital signals. The digital signals representing morphological transforms are stored within a memory element and are

organized as a list of paradigms, wherein each paradigm contains a grouping of one or more of morphological transforms.

Each morphological transform in the paradigm can include a first character string that is stripped from the candidate word and a second string that is added to the character word to morphologically transform the candidate word. Each morphological transform in the paradigm can further include baseform part-of-speech tags and the part-of-speech tag of the morphologically transformed candidate word. These part-of-speech tags aid in identifying appropriate morphological transforms contained within a particular paradigm for application to the candidate word. The morphological analysis system of the invention further provides for a processor capable of stripping character strings and adding character strings to candidate words to form baseforms of variable length.

In accordance with other aspects of the invention, the system provides an addressable memory element having a first addressable table for storing a list of lexical expressions and having a second addressable table for storing a list of paradigms, each paradigm having one or more morphological transforms associated with particular morphological patterns. The lexical expressions stored in the first addressable table of the first memory element can be associated with one or more paradigms listed in the second addressable table.

Further aspects of the invention provide for a data processor having various processing modules. For example, the data processor can include a processing element for matching a morphological transform in an identified paradigm with the candidate word, a processing element for stripping a character string from the candidate word to form an intermediate baseform, and a processing element for adding a character string to the intermediate baseform in accordance with an identified morphological transform.

In accordance with further aspects of the invention, the morphological system provides for identifying a paradigm stored in the memory element equivalent to a candidate word found in a stream of natural language text, matching a morphological pattern in the identified paradigm with the candidate word, and morphologically transforming the candidate word by stripping a first character string from the candidate word and adding a second character string to the candidate word. The morphological system can also identify a paradigm representative of a candidate word found in natural language text by locating a first lexical expression in the first addressable table equivalent to the candidate word and by identifying a paradigm as a function of the located first lexical expression. The association between the first and the second addressable tables allows the identified paradigm to be representative of the candidate word.

Further embodiments of the invention include identifying a part-of-speech tag of the candidate word and matching a morphological pattern in the identified paradigm with the candidate word when the morphological pattern has a part-of-speech tag equivalent to the part-of-speech tag associated with the candidate word. Additional embodiments of the invention include forming an intermediate baseform by stripping a first character string from the candidate word such that the intermediate baseform varies in length as a result of the particular morphological pattern contained within an identified paradigm.

The morphological system can additionally provide for the use of portmanteau paradigms in the second addressable table. The portmanteau paradigms, in comparison to other paradigms, do not necessarily contain inflectional transforms. Rather, the portmanteau paradigms can contain the locations of a plurality of paradigms. The portmanteau paradigm acts as a branching point to other paradigms that contain morphological patterns and morphological transforms. The system thus provides structures and method steps for identifying a plurality of paradigms associated with a lexical expression.

In addition, the portmanteau paradigms can include the location of noun paradigms, verb paradigms, and adjective/adverb paradigms. Accordingly, matching an appropriate morphological paradigm with a candidate word can entail additional steps, which in turn increase the accuracy of morphological transforms. For instance, the matching step can require that the baseform part-of-speech tag associated with a particular morphological pattern match the part-of-speech of the portmanteau paradigm currently under consideration.

Further aspects of the invention include systems for morphologically transforming a candidate word by altering character strings located at any position within the candidate word. For example, the invention transforms digital signals representative of a candidate word by either altering affixes attached to the front, middle, or end of the word (e.g., prefixes, infixes, or suffixes). The invention can accommodate the various locations of affixes by using its unique strip and add algorithm.

The invention provides a system which enables people to enhance the quality of their writing and to use information more effectively. The morphological analyzer and generator is a powerful software tool that hardware and software manufacturers can integrate into applications to help end-users find and retrieve information quickly and easily in multiple languages.

The invention achieves this by providing a linguistically intelligent approach to index pre-processing, search pre-processing, and search expansion. For example, the invention provides a system that identifies the form and formation of words in the source of text, including inflectional and derivational analysis and generation. This allows a database query to be easily expanded to include morphologically related terms. The invention thus improves information retrieval systems by intelligently increasing the recall (i.e., the ratio of relevant items retrieved to the total number of relevant items) and precision (i.e., the ratio of relevant items retrieved to the total number of retrieved items) of index pre-processing, search pre-processing, and search expansion. Additionally, the invention can provide inflectional and derivational analysis and generation to other text-based applications such as dictionaries, thesauruses, and lexicons for spelling correctors and machine-translation systems.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a programmable multilingual text processor according to the present invention;

FIG. 2 illustrates a group of data structures formed by the processor of FIG. 1 according to one practice of the invention;

FIG. 3 shows a word data table utilized by the processor of FIG. 1;

FIG. 4A illustrates a part-of-speech combination table referenced by the word data table of FIG. 3;

FIG. 4B illustrates a suffix table for referencing entries in the part-of-speech combination table of FIG. 4A;

FIG. 4C illustrates a morphological pattern file referenced by the word data table of FIG. 3;

FIG. 5 illustrates possible associations between the tables of FIG. 3, FIG. 4A, and FIG. 4B;

FIG. 6 is a detailed block diagram of a noun-phrase analyzer contained within the text processor of FIG. 1;

FIGS. 7A–7I show flow charts for the tokenizer module illustrated in FIG. 6;

FIG. 8 is a flow chart for the processor shown in FIG. 6;

FIG. 9 is a representative table of rules for the disambiguator shown in FIG. 6;

FIG. 10 illustrates pseudocode for the agreement checker of FIG. 6;

FIG. 11 contains pseudocode for the noun-phrase truncator of FIG. 6;

FIG. 12 illustrates an example of noun-phrase analysis in accordance with the invention;

FIG. 13 contains pseudocode for the morphological analyzer of FIG. 1;

FIG. 14 is a flow chart for the uninflection (inflection reduction) module of FIG. 1;

FIG. 15 is a flow chart for the inflection expansion module of FIG. 1;

FIG. 16 is a flow chart for the underivation (derivation reduction) module of FIG. 1; and

FIG. 17 is a flow chart for the derivation expansion module of FIG. 1.

DETAILED DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates a multilingual text processor 10 in accordance with the invention. The text processor 10 includes a digital computer 12, an external memory 14, a source of text 16, a keyboard 18, a display 20, an application program interface 11, a noun-phrase analyzer 13, and a morphological analyzer/generator 2. Digital computer 12 includes a memory element 22, an input/output controller 26, and a programmable processor 30.

Many of the elements of the multilingual text processor 10 can be selected from any of numerous commercially available devices. For example, digital computer 12 can be a UNIQ 486/33 MHz personal computer; external memory 14 can be a high speed non-volatile storage device, such as a SCSI hard drive; integral memory 22 can be 16 MB of RAM; keyboard 18 can be a standard computer keyboard; and display 20 can be a video monitor. In operation, keyboard 18 and display 20 provide structural elements for interfacing with a user of the multilingual text processor 10. In particular, keyboard 18 inputs user typed commands and display 20 outputs for viewing signal generated by the text processor 10.

The External memory 14 is coupled with the digital computer 12, preferably through the Input/Output Controller 26. Data stored in the External Memory 14 can be downloaded to memory element 22, and data stored in the memory 22 can be correspondingly uploaded to the external memory 14. The external memory 14 can contain various tables utilized by the digital computer 12 to analyze a noun phrase or to perform morphological analysis.

The source of text 16 can be another application program, a keyboard, a communications link, or a data storage device. In either case, the source of text generates and outputs to the digital computer 12 a stream of natural language text. Alternatively, the digital computer 12 may receive as an input from the source of text 16 sentences of encoded text with sentence boundary markers inserted. Sentence splitting per se is known in the art, and is disclosed in Kucera et al., U.S. Pat. No. 4,773,009, entitled Method and Apparatus for Text Analysis. Preferably, the stream of natural language text with identified sentence boundaries enters the digital computer 12 at the Input/Output controller 26.

The Input/Output controller 26 organizes and controls the flow of data between the digital computer 12 and external accessories, such as external memory 14, keyboard 18, display 20, and the source of text 16. Input/Output controllers are known in the art, and frequently are an integral part of standard digital computers sold in the market today.

Application Program Interface 11 includes a set of closely related functions, data types, and operations used in interfacing the computer 12 with the noun-phrase analyzer 13 and with the morphological analyzer/generator 2. In particular, the application program interface 11 comprises four functional elements: App Block, Database Block, Word Block, and Buffer Block. The App Block initiates an application instance, assigns an identification number to it, and passes user processing options to the Noun-phrase Analyzer 13 and to the morphological analyzer/generator 2. The Database Block initializes a database that provides linguistic information about a language. Word Block performs operations on individual words obtained from source text 16, and Buffer Block performs operations on an entire buffer of text obtained from source text 16. Each of the functional elements, i.e., App, Database, Word, and Buffer, contained in interface 11 have associated data structures used to pass information to the noun-phrase analyzer 13 and to the morphological analyzer/generator 2 before processing and to return information from the Application Program Interface 11 after processing by the analyzer 13 and analyzer 2.

The four main functional elements contained in interface 11 perform operations on data structures formed by the application program interface 11. Memory for these functional elements and their associated databases is supplied by the digital computer 12 through the utilization of memory in internal memory element 22 and in external memory element 14.

In operation, App Block is the first functional block called. App Block initiates a session in the noun-phrase analyzer 13 or in the morphological analyzer/generator 2, and assigns a number to the session that uniquely identifies the session. The identifying number is used to track the allocated memory and execution status and to automatically free the memory once the session ends. App Block can start a session to process a single word or an entire buffer of text. In particular, App Block preferably processes one word at a time when the morphological analyzer/generator 2 is called and App Block preferably processes an entire buffer of text when noun-phrase analyzer 13 is called.

Next, Database block is accessed in order to initialize a language database. The language databases provide linguistic information for processing text in a particular language and are used by the noun-phrase analyzer 13 and the morphological analyzer/generator 2. Multiple languages can be processed during any particular session if multiple calls to the database block are made during the session.

After initializing a session by calling App Block and initializing a database by calling Database block, either Word Block or Buffer Block is called, depending on whether a larger amount of text is being processed or one word at a time is being handled. The digital computer 12 fills an input buffer in the application program interface 11 with data from the source text 16, and then calls either Word Block or Buffer Block to begin processing of the text by analyzer 13 or morphological analyzer/generator 2. Following the call, the

analyzer called (i.e., noun-phrase or morphological) scans the input buffer, and creates a stream of tokens in the output buffer and an array that correlates the input and output buffers.

FIG. 1 further illustrates a morphological analyzer/generator 2 that includes an inflection module 4, an uninflection (inflection reduction) module 5, a derivation expansion module 6, and an underivation (derivation reduction) module 7. The inflection module 4 and the uninflection module 5 contain structural features that allow the morphological analyzer/generator 2 to produce all inflected forms of a word given its baseform and to produce all baseforms of a word given an inflection. The derivation expansion module 6 and the underivation module 7 contain features that allow the morphological analyzer/generator 2 to produce all derivatives of a word given its derivational baseform and to produce a derivational baseform of a word given a derivation.

FIG. 2 illustrates the operation of multilingual processor 10. In particular, FIG. 2 shows an input buffer 15, a token list 17, and an output buffer 19. The source of text 16 supplies a stream of natural language text to input/output controller 26 that in turn routes the text to processor 30. Processor 30 supplies the application program interface 11 with the stream of text, and places the text in the input buffer 15. Processor 30 initiates operation of the noun-phrase analyzer 13 by making the calls to the interface 11, as described above.

Noun-phrase analyzer 13 operates upon the text contained in input buffer 15 and generates and places in the interface 11 the token list 17 and the output buffer 19. Token list 17 is an array of tokens that describes the relationship between the input and output data. Token list 17 contains a token 21 for each output word 23. Each token 21 links an input word 25 with its corresponding output word 23 by pointing to both the input word 25 and the output word 23. In addition to linking the input and output, each token describes the words they identify. For example, each token 21 can point to a memory address storing information regarding the particular token. Information associated with each particular token can include the part-of-speech of the token, the capitalization code of the token, the noise-word status of the token, and whether the token is a member of a noun phrase.

In operation, computer 12 obtains a buffer of text from source of text 16, relevant language databases from either the external memory 14 or the internal memory 22, and user selected operations from keyboard 18. Computer 12 then outputs to interface 11 a buffer of text 15, an empty output buffer 19, and the specific operations to be performed on the buffer of text. Noun-phrase analyzer 13 then performs the specified operations on the buffer of text 15 and places the generated output into the output buffer 19 and places the token list 17 that correlates the input buffer of text 15 with the output buffer 19 into the application program interface 11.

FIG. 3 illustrates a word data table 31 used in conjunction with the multilingual text processor 10. Word data table 31 includes digital codings representative of a list of expressions labeled Exp. N_1 through Exp. N_m. The word data table acts as a dictionary of expressions, wherein each expression contains a pointer to an entry, such as the representative entry 33. Various word data tables exist, each being representative of either different languages, dialects, technical language fields, or any subgroup of lexical expressions that can be processed by text processor 30.

The word data table 31 can be an addressable table, such as an 11 byte RAM table stored in a portion of either the external memory 14 or in the memory 12. Each representative entry 33 in the word data table describes the characteristics of one or more words. In particular, entry 33 contains a column, labeled item 35, that describes a particular characteristic of a word. Entry 33 also contains a column, labeled item 37, that identifies which bytes, out of a possible 32-byte prefix position, identify a particular characteristic of the word. For example, particular bytes in the 32-byte prefix position can contain bytes representative of a particular word characteristic, such as the capitalization code of word, or particular bits in the 32-byte prefix position can contain bytes that point to a portion of memory in either memory element 22 or memory element 14 that include information pertaining to a particular characteristic of the word, such as the parts-of-speech of a word.

Characteristics of a word stored in representative entry 33 include the part-of-speech combination index of a word, and the grammatical features of the word. In particular the part-of-speech combination index of a word is identified by the labeled field 44 in FIG. 3, while the grammatical features of the word are identified by the labeled fields 32, 34, 36, 38, 40, 42, 46, 48, 50, 52, 54, 56, 58, and 60 in FIG. 3. Additional grammatical features of a word include the word length, the language code, whether the word is an abbreviation, and whether the word is a contraction. Although not shown in FIG. 3, addresses to these additional grammatical features of a word can be stored in a representative entry 33. For example, positions 12–13 in the 32-byte prefix location can identify the word length; positions 1–2 in the 32-byte prefix location can identify the language code; position 19 can indicate whether the word is an abbreviation; and position 20 can indicate whether the word is a contraction. The preferred implementation is for the byte values in the 32-byte prefix to be encoded in a compressed form.

The Capcode Field 32 identifies the capitalization of the word. For example, Capcode Field 32 can store a binary number representative of the capitalization characteristics of the word, such as: "000" can represent all lowercase letters; "001" can represent initial letter uppercase; "010" can represent all uppercase letters; "011" can represent the use of a capitalization map (mixed capitalization); "100" can represent no capitalization, unless the word is located at the beginning of a sentence; and "101" can represent that capitalization is not applicable.

The Dialect Field 34 is used to identify words properly spelled in one dialect, but improperly spelled in another dialect. A common example of this behavior can be demonstrated using the American term color and its British counterpart colour. This field is generally accessed during the decoding process to filter words based on the dialect of the word.

The Has Mandatory Hyphen Field 36 stores information about words which change spelling when hyphenated at the ends of lines. In Germanic languages, the spelling of a word may change if it is hyphenated. This information can be encoded for both the hyphenated and unhyphenated forms of a word. The presence or absence of the hyphen at the Error Position is enough to identify whether the word is correctly or incorrectly spelled. An example is the German word bak-ken, which is the form of the word used when it is hyphenated; without the hyphen, the word is spelled backen. This information links the hyphenated form with its unhyphenated form which would be the form normally used for such information retrieval tasks as indexing.

The Is Derivation Field 38 is used to identify whether a word is a derivation (i.e., is a derived form of a root and

therefore should use the derivation pattern to find the root form) or a derivational root (in which case the derivation pattern is used to produce the derived forms of the root). For example, the word readable is a derived form of the derivational root read.

The Restricted/Word-Frequency Field **40** is used to store the word-frequency information about words in the word data table.

The POS Combination Index Field **44** stores an index into the part-of-speech combination table **62**, as illustrated in FIG. **4**. The part-of-speech combination table contains a list of parts-of-speech that a word can take. The parts-of-speech are stored with the most frequent part-of-speech tag listed first in the part-of-speech combination table. The order of the other parts-of-speech in this table is unspecified, but implied to be in reverse frequency order. English lists about 650 entries in this table, French about 1900, Swedish about 2000. Other languages fall within this range.

The Noun Inflection Pattern Field **46**, the Verb Inflection Pattern Field **48**, and the Adjective/Adverb Inflection Pattern Field **50** give the respective pattern numbers used in inflecting or uninflecting noun, verb, and adjective/adverb forms. The pattern number indexes a separate table of inflectional endings and their parts-of-speech. Thus, there is an index to the noun inflection pattern of the word, an index to the verb inflection pattern of the word, and an index to the inflection pattern representative of the inflections of both the adjective and adverbial forms of the word.

The Derivation Pattern Field **52** contains information about how to derive or underive words from this particular word. Derivation patterns are much like inflection patterns. The derivation pattern is an index into a table of derivational endings and their parts-of-speech. The Is Derivation Field **38** described above tells whether the pattern should be used for deriving or underiving. If the bit contained within the Is Derivation Field **38** is not set, the word is a derivational root.

The Compound Info Field **54** indexes another lookup table identifying rules regarding the compounding characteristics of the word. The lookup table contains fields, including a left-most compound component, a right-most compound component, that identify possible positions where the word can be used as a component in a compound word. This information is used for Germanic languages to decompose compounds into their constituents. For example, the German compound Versicherungsgesellschaft (insurance company) can be decomposed into Versicherung (its left-most compound component) and Gesellschaft (its right-most compound component).

The Error Position Field **56** specifies the position of a spelling-changing hyphen.

The LMCC Link Length Field **58** specifies the length of the compound link and is only used for words marked as being a Left-Most Compound Component. In the example above, the left-most compound component Versicherung has a Link Field of 1 since the single character s is used as its compound link.

The Field of Interest Field **60** describes the topic or domain of the given entry. For example, field **60** can differentiate terms used exclusively in Medicine from those that are used exclusively in Law.

FIGS. **4A**, **4B**, and **4C** illustrate other tables used by the multilingual text processor and stored in portions of either external memory **14** or internal memory **22**. In particular, FIG. **4A** shows a Part-of-Speech Combination Table **62** containing a list of indexes **64**, a list of part-of-speech tags **66**, and a list of OEM tags **68**; and FIG. **4B** shows a Suffix Table **70** having a list of suffixes **72** and having a list of POS indexes **74** to the part-of-speech combination table **62**; and FIG. **4C** shows a morphological file **71** having a list of paradigm umbers **73** each having a list of associated transformations identified by columns **75**, **77** and **79**.

These tables can be modified according to particular languages, such that the tables can provide linguistic information for processing text in a particular language. Text processing system **10** can load tables associated with particular language databases when the database block of the application program interface **11** is initialized. This advantageously allows the databases to change without affecting the source code of the application program interface **11**, the noun-phrase analyzer **13**, or the morphological analyzer/generator **2**. Thus, in effect the source code becomes independent of the language being processed. Further in accordance with this invention, multiple languages can be processed by creating a database instance for each language being processed. The languages can be selected from either English, German, Spanish, Portuguese, French, Dutch, Italian, Swedish, Danish, Norwegian, or Japanese. These particular languages are representative of languages having their own specific rules and tables for analyzing noun phrases, but are not included as a limitation of the invention.

As shown in FIG. **4A**, each entry in part-of-speech combination table **62** contains an index **64** having one or more associated part-of-speech tags **66** and having an associated, simpler OEM part-of-speech tag **68** used for display to users of the system. Each index **64** in table **62** identifies one or more part-of-speech tags **66**. Thus, all words contained within the word data table are associated with one or more part-of-speech tag **66**. If the part-of-speech tag entry **66** includes multiple part-of-speech tags, the most probable tag is the first tag in the entry **66**. For example, as illustrated in FIG. **4A**, if the Index **64** of a word is 1, the word has a single part-of-speech tag **66** of NN (used to identify generic singular nouns); and if the Index **64** of a word is 344, the word has five possible part-of-speech tags. Furthermore, a word indexed to 344 in the combination table has a most probable part-of-speech tag of ABN (used to identify pre-qualifiers such as half and all), and also has part-of-speech tags of NN (used to identify generic singular nouns), NNS (used to identify generic plural nouns), QL (used to identify qualifying adverbs), and RB (used to identify generic adverbs).

FIG. **4B** illustrates a Suffix table **70** having a list of suffixes **72** and having a list of POS indexes **74** to the part-of-speech combination table **62**. Thus, each entry in table **70** has a suffix **72** associated with a POS index **74**. In operation, the suffix of a word contained in a stream of text can be compared with suffix entries **72** in table **70**. If a match is found for the suffix of the extracted word, then the word can be associated with a part-of-speech tag **66** in part-of-speech table **62** through POS index **74**. For example, if a word in the stream of text contains a suffix, le (as in ô'le), that word can be identified in table **70** and be associated with a part-of-speech index "001". The part-of-speech index "001" contains a part-of-speech tag NN (noun), as illustrated in FIG. **4A**. Similarly, the word in the stream of text having a suffix am (as in m'am) can be associated with a part-of-speech tag of NN through tables **62** and **70**.

FIG. **4C** illustrates an exemplary morphological file **71** where each horizontal line shown in the morphological file **71** is a separate morphological paradigm having one or more morphological transforms. Vertical column **73** identifies the numbering of the morphological paradigms, and columns **75**, **77**, and **79** identify vertical columns containing different

morphological transforms associated with any particular morphological paradigm. Each morphological transform is formed of a plurality of functional elements. In operation, the morphological file 71 of FIG. 4C describes how to produce a morphological transform given a baseform.

The morphological transforms identified by columns 75, 77 and 79 are all similarly structured. For example, each transform contains at least two functional elements that indicate one character string to be removed and one character string to be added to a candidate word. The similarity between the transforms allows processor 30 to uniformly apply the functional elements contained in any particular transform without having to consider exceptions to a discrete set of standard rules. The uniformity in the actions of processor 30, regardless of the transform being considered, allows for quick and easy processing.

As shown in FIG. 4C, every morphological transform identified in columns 75, 77 and 79 is structured as follows:

baseform part-of-speech tag _____ first character string to strip from the candidate word →
second character string to add to the candidate word_____ part-of-speech tag of morphological transform
[optional field for prefixation].

Each morphological transform can thus be described as containing a number of functional elements listed in sequence, as shown in FIG. 4C. In particular, the first functional element specifies the part-of-speech tag of the baseform of the candidate word, and the second functional element identifies the suffix to strip from the candidate word to form an intermediate baseform. The third functional element identifies the suffix to add to the intermediate baseform to generate the actual baseform, and the fourth functional element specifies the part-of-speech of the morphological transform. The fifth functional element is an optional element indicating whether prefixation occurs.

FIG. 4C illustrates, in particular, a morphological file suited to inflection and uninflection. For example, inflection transform 001 (as identified by column 73) contains three transformations shown in columns 75, 77 and 79, respectively. The column 75 transformation for inflection transform 001 contains the transform, VB_→d_VBN. This transform contains rules specifying that: (1) the baseform part-of-speech is VB; (2) no suffix is to be stripped from the candidate word to form the intermediate baseform; (3) the suffix d is to be added to the intermediate baseform to generate the actual baseform; (4) the part-of-speech of the resulting inflected form is VBN; and (5) no prefixation occurs. The column 79 transformation for transform 001 contains the transform VB_e→ing_VBG. This transform specifies: (1) the baseform part-of-speech is VB; (2) the suffix e is to be stripped from the candidate word to form the intermediate baseform; (3) the suffix ing is to be added to the intermediate baseform to generate the actual baseform; (4) the part-of-speech of the resulting inflected form is VBG; and (5) no prefixation occurs.

A file similar to that shown in FIG. 4C can be constructed for derivation expansion and underivation (derivation reduction). A derivational file, however, will not contain a functional element in the transform identifying part-of-speech information used in specifying whether a candidate word is a derivation or a derivational baseform. Information regarding derivation baseforms is instead stored in the word data table 31 of FIG. 3 under the Is Derivation Field 38.

Morphological file 71 of FIG. 4C also illustrates the use of portmanteau paradigms. Portmanteau paradigms provide a structure capable of mapping the morphological changes associated with words having complicated morphological patterns. In particular, morphological transforms 133, 134, 135, 136 and 137 (as identified in column 73) contain portmanteau paradigm used for associating a plurality of paradigms with any particular candidate word.

Morphological transform 133 indicates that patterns "006" and "002", as identified in column 73, are used to inflect the candidate word associated with morphological transform 133. Accordingly, a candidate word associated with inflection transform 133 becomes further associated with inflection transforms 002 and 006. For instance, the portmanteau paradigm 133 identifies the two inflections of travel, that can be inflected as travelled and traveled, depending upon dialect. Portmanteau paradigm 133 can also be used to inflect install, which can also be spelled instal. The illustrated portmanteau paradigms illustrate one possible structure used for applying multiple paradigms to any particular candidate word.

Another possible structure for providing portmanteau paradigms can be formed using word data table 31 and a representative entry 33, as shown in FIG. 3. For example, expression N_2 in data table 31 points to a representative entry 33 having a noun inflection pattern 46, a verb inflection pattern 48, and an adjective/adverb inflection pattern 50. In addition, the patterns 46, 48, and 50 each point to a paradigm in a morphological file 71, as illustrated in FIG. 4C. Thus, a candidate word matched with the expression N_2 can become associated with a plurality of paradigms.

FIG. 4C illustrates a further aspect of the invention wherein the applicants' system departs dramatically from the prior art. In particular, a morphological baseform in accordance with the invention can vary in length and does not need to remain invariant. By utilizing baseforms of variable length, the invention removes many of the disadvantages associated with earlier natural language processing techniques, including the need for a large exception dictionary.

The morphological file 71 includes transforms having a variable length baseform, such as paradigm numbers 001 and 004. For example, the column 75 and 77 transforms of paradigm 001 produce a baseform having no characters removed from the candidate word while the column 79 transform of paradigm 001 produces a baseform having an e character removed. The column 75 transform of paradigm 004 produces a baseform having no characters removed while the column 77 and 79 transforms of paradigm 004 produce baseforms having ay character removed from the candidate word. Thus, when processor 30 acts in accordance with the instructions of paradigms 001 or 004 to form all possible baseforms of a candidate word, the processor will form baseforms that vary in length.

FIG. 5 illustrates a database system stored in various portions of memory elements 14 and 22 showing a connection between tables 31, 62, and 70 for associating part-of-speech tags with various lexical expressions contained within a stream of text. An Expression N_2 contained within

the stream of text can be identified in the word data table **31** as representative entry **33**. Representative entry **33** encodes the information contained in a 32-byte prefix, of which bytes **16–18** contain a code found in the part-of-speech combination table **62**. This table in its turn relates this particular part-of-speech combination with index **343** in table **62**, thereby associating the part-of-speech tags of ABN (prequalifier), NN (noun), QL (qualifying adverb), and RB (adverb) with Expression N_2.

In accordance with a further aspect of the invention, a part-of-speech tag can be associated with an expression in the stream of text through the use of suffix table **70**. For example, a first expression in stream of text might contain a suffix ôle, and can be identified in suffix table **70** as representative entry **63**. A second expression in the stream of text might contain the suffix ôle, and can be identified in suffix table **70** as representative entry **65**. The pointer in representative entry **63** points to index **1** in table **62**, and the pointer in representative entry **65** points to index **1** in table **62**. Thus, both the first and second expression in the stream of text become associated with the part-of-speech tag of NN.

FIG. **6** shows a block diagram of a noun-phrase analyzer **13** for identifying noun phrases contained within a stream of natural language text. The analyzer **13** comprises a tokenizer **43**, a memory element **45**, and a processor **47** having: a part-of-speech identifier **49**, a grammatical feature identifier **51**, a noun-phase identifier **53**, an agreement checker **57**, a disambiguator **59**, and a noun-phrase truncator **61**. Internal connection lines are shown both between the tokenizer **43** and the processor **47**, and between the memory element **45** and the processor **47**. FIG. **6** further illustrates an input line **41** to the tokenizer **43** from the application program interface **11** and an output line from the processor **47** to the application program interface **11**.

Tokenizer **43** extracts tokens (i.e., white-space delimited strings with leading and trailing punctuation removed) from a stream of natural language text. The stream of natural language text is obtained from text source **16** through the application program interface **11**. Systems capable of removing and identifying white-space delimited strings are known in the art and can be used herein as part of the noun-phrase analyzer **13**. The extracted tokens are further processed by processor **47** to determine whether the extracted tokens are members of a noun phrase.

As illustrated in FIGS. **7A–7I**, tokenizer **43** can comprise a system for extracting lexical matter from the stream of text and a system for filtering the stream of text. Tokenizer **43** receives input from input line **41** in the form of a text stream consisting of alternating lexical and non-lexical matter; accordingly, lexical tokens are separated by non-lexical matter. Lexical matter can be broadly defined as information that can be found in a lexicon or dictionary, and is relevant for Information Retrieval Processes. Tokenizer **43** identifies the lexical matter as a token, and assigns the attributes of the token into a bit map. The attributes of the non-lexical matter following the lexical token are mapped into another bit map and associated with the token. Tokenizer **43** can further tag or identify those tokens that are candidates for further linguistic processing. This filtering effect by the tokenizer **43** reduces the amount of data processed and increases the overall system throughput.

This implementation of tokenizer **28** has several benefits. It achieves high throughput; it generates information about each token during a first pass across the input stream of text; it eliminates and reduces multiple scans per token; it does not require the accessing of a database; it is sensitive to changes in language; and it generates sufficient information to perform sophisticated linguistic processing on the stream of text. Moreover, tokenizer **28** allows the non-lexical matter following each token to be processed in one call. Additionally, tokenizer **28** achieves these goals while simultaneously storing the properties of the non-lexical string in less space than is required to store the actual string.

Memory element **45**, as illustrated in FIG. **5**, can be a separate addressable memory element dedicated to the noun-phrase analyzer **13**, or it can be a portion of either internal memory element **22** or external memory element **14**. Memory element **5** provides a space for storing digital signals being processed or generated by the tokenizer **43** and the processor **47**. For example, memory element **14** can store tokens generated by tokenizer **43**, and can store various attributes identified with a particular token by processor **47**. In another aspect of the invention, memory element **14** provides a place for storing a sequence of tokens along with their associated characteristics, called a window of tokens. The window of tokens is utilized by the processor to identify characteristics of a particular candidate token by evaluating the tokens surrounding the candidate token in the window of extracted tokens.

Processor **47**, as illustrated in FIG. **6**, operates on the extracted tokens with various modules to form noun phrases. These modules can be hard-wired digital circuitry performing functions or they can be software instructions implemented by a data processing unit performing the same functions. Particular modules used by processor **47** to implement noun-phrase analysis include modules that: identify the part-of-speech of the extracted tokens, identify the grammatical features of the extracted tokens, disambiguate the extracted tokens, identify agreement between extracted tokens, and identify the boundaries of noun phrases.

FIG. **8** depicts a processing sequence of noun-phrase analyzer **13** for forming noun phrases that begins at step **242**. At step **243**, the user-specified options are input to the noun-phrase analysis system. In particular, those options identified by the user through an input device, such as keyboard **18**, are input to text processor **10** and channeled through the program interface **11** to the noun-phrase analyzer **13**. The user selected options control certain processing steps within the noun-phrase analyzer as detailed below. At step **244**, the user also specifies the text to be processed. The specified text is generally input from source text **16**, although the text can additionally be internally generated within the digital computer **12**. The specified text is channeled through the application program interface **11** to the noun-phrase analyzer **13** within the Buffer Block. Logical flow proceeds from box **244** to box **245**.

At action box **245** tokenizer **43** extracts a token from the stream of text specified by the user. In one embodiment, the tokenizer extracts a first token representative of the first lexical expression contained in the stream of natural language text and continues to extract tokens representative of each succeeding lexical expression contained in the identified stream of text. In this embodiment, the tokenizer continues extracting tokens until either a buffer, such as memory element **45**, is full of the extracted tokens or until the tokenizer reaches the end of the text stream input by the user. Thus, in one aspect the tokenizer extracts tokens from the stream of text one token at a time while in a second aspect the tokenizer tokenizes an entire stream of text without interruption.

Decision box **246** branches logical control depending upon whether or not three sequential tokens have been extracted from the stream of text by tokenizer **43**. At least three sequential tokens have to be extracted to identify noun

phrases contained within the stream of text. The noun-phrase analyzer **13** is a contextual analysis system that identifies noun phrases based on a window of token containing a candidate token and at least one token preceding the candidate token and one token following the candidate token in the stream of text. If at least three tokens have not yet been extracted, control branches back to action box **245** for further token extraction, while if three tokens have been extracted logical flow proceeds to decision box **247**.

At decision box **247** the system identifies whether the user-requested disambiguation of the part-of-speech of the tokens. If the user has not requested part-of-speech disambiguation control proceeds to action box **249**. If the user has requested part-of-speech disambiguation, the logical control flow proceeds to decision box **248** wherein the system determines whether or not disambiguation can be performed. The noun-phrase analyzer **13** disambiguates tokens within the stream of natural language text by performing further contextual analysis. In particular, the disambiguator analyzes a window of at most four sequential tokens to disambiguate part-of-speech of a candidate token. In one aspect the window of token contains the two tokens preceding an ambiguous candidate token, the ambiguous candidate token itself, and a token following the ambiguous candidate token in the stream of text. Thus, in accordance with this aspect, if four sequential tokens have not been extracted logical flow branches back to action box **245** to extract further tokens from the stream of text, and if four sequential tokens have been extracted from the stream of text logical flow proceeds to action box **249**.

At action box **249**, the part-of-speech identification module **49** of processor **47** determines the part-of-speech tags for tokens extracted from the stream of text. The part-of-speech tag for each token can be determined by various approaches, including: table-driven, suffix-matching, and default tagging methods. Once a part-of-speech tag is determined for each token, the part-of-speech tag becomes associated with each respective token. After step **249**, each token **21** in token list **17** preferably contains the most probable part-of-speech tag and contains a pointer to an address in a memory element containing a list of other potential part-of-speech tags.

In accordance with the table driven aspect of the invention, the part-of-speech tag of a token can be determined using the tables shown in FIGS. **3–5**. For example, a representative lexical expression equivalent to the extracted token can be located in the word data table **31** of FIG. **2**. As shown in FIG. **2**–FIG. **5**, module **49** can then follow the pointer, contained in bytes **16–18** of the representative expression in word table **31**, to an index **64** in the part-of-speech combination table **62**. The index **64** allows module **49** to access a field **66** containing one or more part-of-speech tags. Module **49** at processor **47** can then retrieve these part-of-speech tags or store the index to the part-of-speech tags with the extracted token.

This table-driven approach for identifying the part-of-speech tags of extracted words advantageously provides a fast and efficient way of identifying and associating parts-of-speech with each extracted word. The word data table and the POS Combination Table further provide flexibility by providing the system the ability to change its part-of-speech tags in association with the various language databases. For example, new tables can be easily downloaded into external memory **14** or memory **22** of the noun-phrase system without changing any other sections of the multilingual text processor **10**.

In accordance with the suffix-matching aspect of the invention, the part-of-speech tag of a token can be determined using the tables shown in FIGS. **4–5**. For example, module **49** at processor **47** can identify a representative suffix consisting of the last end characters of the extracted token in suffix table **70** of FIG. **4B**. Once a matching suffix is identified in suffix table **70**, module **49** can follow the pointer in column **74** to an index **64** in part-of-speech combination table **62**. The index **64** allows module **49** to access a field **66** containing one or more part-of-speech tags. The index **64** allows module **49** to access a field **66** containing one or more part-of-speech tags. The part-of-speech identification module **49** can then retrieve these part-of-speech tags or store the index to the part-of-speech tags with the extracted token. Generally, the suffix-matching method is applied if no representative entry in the word data table **31** was found for the extracted token.

A second alternative method for identifying the part-of-speech tags for the token involves default tagging. Generally, default tagging is only applied when the token was not identified in the word data table **31** and was not identified in suffix table **70**. Default tagging associates the part-of-speech tag of NN (noun) with the token. As a result, at the end of step **249** each token has a part-of-speech tag or part-of-speech index that in turn refers to either single or multiple part-of-speech tags. After step **249**, logical control flows to action box **250**.

At action box **250**, the grammatical feature identification module **51** of the processor **9** determines the grammatical features for the tokens **21** contained in the token list **17**. The grammatical features for each token can be obtained by identifying a representative entry for the token in the word data table **31** of FIG. **3**. The identified representative entry contains information pertaining to the grammatical features of the word in fields **32, 34, 36, 38, 40, 42, 46, 48, 50, 52, 54, 56, 58** and **60**. These fields in the representative entry either contain digital data concerning the grammatical features of the token, or point to an address in a memory element containing the grammatical features of the token. After box **250**, control proceeds to decision box **251**.

Decision box **251** queries whether the user requested disambiguation of the part-of-speech tags. If disambiguation was requested, control proceeds to action box **252**. If disambiguation was not requested, control proceeds to action box **253**. At action box **252**, the part-of-speech tags of ambiguous tokens are disambiguated.

The disambiguator module **59** of the processor **47** identifies tokens having multiple part-of-speech tags as ambiguous and disambiguates the identified ambiguous tokens. Accordingly, action box **252** disambiguates those tokens identified as having multiple part-of-speech tags. For example, a first token extracted from the stream of text can be identified in the word data table **31** and thereby have associated with the first token an index **64** to the part-of-speech combination table **62**. Furthermore, this index **64** can identify an entry having multiple part-of-speech tags in column **66** of table **62**. Thus, the first token can be associated with multiple part-of-speech tags and be identified as ambiguous by processor **47**.

Preferably, the first listed part-of-speech tag in table **62**, called a primary part-of-speech tag, is the part-of-speech tag having the highest probability of occurrence based on frequency of use across different written genres and topics. The other part-of-speech tags that follow the primary part-of-speech tag in column **66** of table **62** are called the secondary part-of-speech tags. The secondary part-of-speech tags are so named because they have a lower probability of occurrence than the primary part-of-speech tag. The disambiguator can choose to rely on the primary part-of-speech tag as

the part-of-speech tag to be associated with the ambiguous token. However, to ensure accurate identification of the part-of-speech for each token, this probabilistic method is not always reliable. Accordingly, in a preferred aspect, the invention provides for a disambiguator module **59** that can disambiguate those tokens having multiple part-of-speech tags through contextual analysis of the ambiguous token.

In particular, disambiguator **59** identifies a window of sequential tokens containing the ambiguous token and then determines the correct part-of-speech tag as a function of the window of sequential tokens. In a first embodiment, the window of sequential tokens can include, but is not limited to, the two tokens immediately preceding the ambiguous token and the token immediately following the ambiguous token. In a second embodiment, the window of sequential tokens includes the ambiguous token, but excludes those classes of tokens not considered particularly relevant in disambiguating the ambiguous token. One class of tokens considered less relevant in disambiguating ambiguous tokens include those tokens having part-of-speech tags of either: adverb; qualifying adverb; or negative adverbs, such as never and not. This class of tokens is collectively referred to as tokens having "ignore tags". Under the second embodiment, for example, the disambiguator module **59** forms a window of sequential tokens containing will run after skipping those words having ignore tags in the following phrases: will run; will frequently run; will very frequently run; will not run; and will never run. The second embodiment thus ensures, by skipping or ignoring a class of irrelevant tokens, an accurate and rapid contextual analysis of the ambiguous token without having to expand the number of tokens in the window of sequential tokens. Moreover, a window of four sequential tokens ranging from the two tokens immediately preceding the ambiguous token and the token immediately following the ambiguous token can be expanded to include additional tokens by: (1) skipping those tokens contained within the original window of four sequential tokens that have ignore tags, and (2) replacing the skipped tokens with additional sequential tokens surrounding the ambiguous token.

The functions or rules applied by module **59** identify the most accurate part-of-speech of the ambiguous token based both upon the window of sequential tokens containing the ambiguous token and the characteristics associated with those tokens contained within the window of tokens. The characteristics associated with the tokens include, either separately or in combination, the part-of-speech tags of the tokens and the grammatical features of the tokens.

Once the disambiguator module **59** of the processor **47** has identified the most accurate part-of-speech tag, the processor places this part-of-speech tag in the position of the primary part-of-speech tag, i.e., first in the list of the plurality of part-of-speech tags associated with the ambiguous token. Thus, the ambiguous target token remains associated with a plurality of part-of-speech tags after the operations of processor **47**, but the first part-of-speech tag in the list of multiple part-of-speech tags has been verified as the most contextually accurate part-of-speech tag for the ambiguous token.

In one aspect, disambiguator **59** can determine that no disambiguation rules apply to the ambiguous token and can thus choose to not change the ordering of the plurality of part-of-speech tags associated with the ambiguous token. For example, a token having multiple part-of-speech tags has at least one part-of-speech tag identified as the primary part-of-speech tag. The primary part-of-speech tag can be identified because it is the first part-of-speech tag in the list of possible part-of-speech tags, as illustrated in FIG. **4A**. If the disambiguator **59** determines that no disambiguation rules apply, the primary part-of-speech tag remains the first part-of-speech tag in the list.

In a further aspect, a disambiguation rule can be triggered and one of the secondary part-of-speech tags can be promoted to the primary part-of-speech tag. In accordance with another aspect, a disambiguation rule is triggered and the primary part-of-speech tag of the ambiguous token is coerced into a new part-of-speech tag, not necessarily found amongst the secondary part-of-speech tags. An additional aspect of the invention provides for a method wherein a disambiguation rule is triggered but other conditions required to satisfy the rule fail, and the primary part-of-speech tag is not modified. Thus, after disambiguating, each token has a highly reliable part-of-speech tag identified as the primary part-of-speech tag.

FIG. **9** illustrates an exemplary rule table used for disambiguating an extracted token in the English language. As discussed with respect to the tables illustrated in FIG. **3**–FIG. **5**, the disambiguation tables can differ from language to language. Advantageously, the tables can be added to the system **10** or removed from the system **10** to accommodate various languages without modifying the source code or hardware utilized in constructing the multilingual text processor **10** in accordance with the invention.

The illustrated table contains: (1) a column of rules numbered **1**–**6** and identified with label **261**; (2) a column representing the ambiguous token [i] and identified with label **264**; (3) a column representing the token [i+1] immediately following the ambiguous token and identified with label **266**; (4) a column representing the token [i–1] immediately preceding the ambiguous token and identified with the label **262**; and (5) a column representing the token [i–2] immediately preceding the token [i–1] and identified with the label **260**. Accordingly, the table illustrated in FIG. **9** represents a group of six disambiguation rules that are applied by disambiguator **59**, as part of the operations of the processor **47**, to a window of sequential tokens containing the ambiguous token [i]. In particular, each rule contains a set of requirements in columns **260**, **262**, **264**, and **266**, which if satisfied, cause the primary part-of-speech of the ambiguous token to be altered. In operation, processor **47** sequentially applies each rule to an ambiguous token in the stream of text and alters the primary part-of-speech tag in accordance with any applicable rule contained within the table.

For example, rule **1** has a requirement and result labeled as item **268** in FIG. **9**. In accordance with rule **1**, the processor **47** coerces the primary part-of-speech tag of the ambiguous token to NN (singular common noun) if the ambiguous token [i] is at the beginning of a sentence and has a Capcode greater than 000 and does not have a part-of-speech tag of noun.

Rules **2**–**6**, in FIG. **9**, illustrate the promotion of a secondary part-of-speech tag to the primary part-of-speech tag as a function of a window of token surrounding the ambiguous token [i]. In particular, rule **2** promotes the secondary part-of-speech tag of singular common noun to the primary part-of-speech tag if: the token [i–2] has a primary part-of-speech tag of article, as shown by entry **270**; the token [i] has a primary part-of-speech tag of either verb or second possessive pronoun or exclamation or verb past tense form, as shown by entry **272**; and the token [i] has a secondary part-of-speech tag of singular common noun, as shown by entry **272**. Rule **3** promotes the secondary part-of-speech tag of singular common noun to the primary part-of-speech tag if: the token [i–1] has a part-of-speech tag of verb infinitive

or singular common noun, as shown by entry **274**; and the token |i| has a primary part-of-speech tag of verb or second possessive pronoun or exclamation or verb past tense form and has a secondary part-of-speech tag of singular common noun, as shown by entry **276**. Rule 4 promotes the secondary part-of-speech tag of singular common noun to the primary part-of-speech tag if: the token |i−1| has a part-of-speech tag of modal auxiliary or singular common noun, as shown by entry **278**; the token |i| has a primary part-of-speech tag of modal auxiliary and has a second part-of-speech tag of singular common noun, as shown by entry **280**; and the token |i+1| has a part-of-speech tag of infinitive, as shown by entry **282**.

FIG. **9** thus illustrates one embodiment of the invention wherein the disambiguator **59** of the processor **47** modifies the ambiguous target token in accordance with a rule table. In particular, the illustrated rule table instructs processor **47** to modify the part-of-speech tags of the ambiguous token as a function of: the two tokens preceding the ambiguous target token in the stream of text, the token following the ambiguous target token in the stream of text, and the ambiguous target token itself. FIG. **9** further illustrates an embodiment wherein the ambiguous target token is modified as a function of the primary part-of-speech tag and the secondary part-of-speech tags of the ambiguous target token, and the part-of-speech tags of the other token surrounding the target token.

Disambiguation step **252** can also provide for a system that aids in identifying the elements of a noun phrase by checking whether or not the tokens in the stream of natural language text agree in gender, number, definiteness, and case. In particular, processor **47** can validate agreement between a candidate token and a token immediately adjacent (i.e., either immediately preceding or immediately following) the candidate token in the stream of text.

Agreement analysis prior to step **253**, wherein the noun phrase is identified, operates in a single match mode that returns a success immediately after the first successful match. Thus, if agreement is being tested for token [i] and token [i−1] in the single match mode, processing stops as soon as a match is found. In accordance with this process, the processor selects the first part-of-speech tag from token [i], and tries to match it with each tag for the token [i−1] until success is reached or all of the part-of-speech tags in token [i−1] are exhausted. If no match is found, then the processor **47** tries to match the next part-of-speech tag in the token [i] with each tag in token [i−1] until success is reached or all of the part-of-speech tags in token [i−1] are exhausted. This process continues until either a match is reached, or all of the part-of-speech tags in both token [i] and token [i−1] have been checked with each other. A successful agreement found between two tokens indicates that the two tokens are to be treated as part of a noun phrase. If no agreement is found, then the two tokens are not considered to be a part of the same noun phrase.

First, the first POS tag from each token in checked for agreement.

	Agreement Tags	Agreement Tags	Agreement Tags
i−1	Plural, Masculine	Singular, Masculine	
i	Singular, Feminine	Singular, Masculine	Plural, Masculine
(Tag1 & Tag2 & Number Map) fails		&	(Tag1 & Tag2 & GenderMap) fails

If this fails, the second POS tag from the token [i−1] is checked for a match:

	Agreement Tags	Agreement Tags	Agreement Tags
i−1	Plural, Masculine	Singular, Masculine	
i	Singular, Feminine	Singular, Masculine	Plural, Masculine
(Tag1 & Tag2 & Number Map) passes		&	(Tag1 & Tag2 & GenderMap) fails

At this point, all of the POS maps in the token |i−1| have been exhausted, and no successful match has been bound. The second POS tag in the token |i| must now be compared with all of the POS tags in the token |i−1|.

The first POS tag from the token |i−1| and the second tag from the token [i] are checked for a match:

	Agreement Tags	Agreement Tags	Agreement Tags
i−1	Plural, Masculine	Singular, Masculine	
i	Singular, Feminine	Singular, Feminine	Plural, Masculine
(Tag1 & Tag2 & Number Map) fails		&	(Tag1 & Tag2 & GenderMap) passes

If it fails, the second POS tag from the token [i−1] is checked for agreement:

	Agreement Tags	Agreement Tags	Agreement Tags
i−1	Plural, Masculine	Singular, Masculine	
i	Singular, Feminine	Singular, Masculine	Plural, Masculine
(Tag1 & Tag2 & Number Map) passes		&	(Tag1 & Tag2 & GenderMap) passes

At this point, a match has successfully been made, and all agreement processing stops. The two tokens agree and Single Match mode processing is complete.

After Step **252**, logical flow proceeds to Step **253**. At step **253**, the noun-phrase identifier module **53** of processor **47** identifies the boundaries of noun phrases contained within the stream of natural language text, and marks those tokens forming the noun phrase. In particular, processor **47** identifies the noun-phrase boundaries through contextual analysis of each extracted token in the stream of text. In addition, module **53** marks those tokens forming the noun phrase by tagging tokens contained within the noun phrase. For example, module **53** can associate with: the first token in the noun phrase a tag indicating "the beginning" of the noun phrase; the last token in the noun phrase a tag indicating "the end" of the noun phrase; and those tokens found between the first and last tokens in the noun phrase a tag indicating "the middle" of the noun phrase. Thus, module **53** of processor **47** identifies those tokens that it determines are members of a noun phrase as either "the beginning", "the middle", or "the end" of the noun phrase.

According to one aspect of the invention, the noun-phrase identifier module **53** processor **47** forms a window of sequential tokens to aid in identifying members of a noun phrase. Further in accordance with this aspect, the window of sequential tokens includes a token currently undergoing analysis, called a candidate token, and tokens preceding and following the candidate token in the stream of text. Preferably, the window of tokens includes the candidate token and one token immediately following the candidate token in the stream of text and one token immediately preceding the candidate token in the stream of text. Thus, the window contains at least three extracted tokens ranging from

the token preceding the candidate token to the token following the candidate token inclusive. This window of sequential tokens provides a basis for contextually analyzing the candidate token to determine whether or not it is a member of a noun phrase.

The module 53 analyses characteristics of the window of sequential tokens to determine whether the candidate token is a member of a noun phrase. The characteristics analyzed by processor 47 include, either separately or in conjunction, the part-of-speech tags and the grammatical features of each of the tokens contained within the window of tokens. Module 53 of processor 47 contextually analyzes the candidate token by applying a set of rules or functions to the window of sequential tokens surrounding the candidate token, and the respective characteristics of the window of sequential tokens. By applying these rules, module 53 identifies those candidate tokens which are members of noun phrases contained within the stream of text.

The noun-phrase identification rules are a set of hard-coded rules that define the conditions required to start, continue, and terminate a noun phrase. In general, noun phrases are formed by concatenating together two or more contiguous tokens having parts-of-speech functionally related to nouns. Those parts-of-speech functionally related to nouns include the following parts-of-speech: singular common noun (NN), adjective (JJ), ordinal number (ON), cardinal number (CD). In one embodiment, the noun-phrase rules apply these concepts and form noun phrases from those sequential tokens having parts-of-speech functionally related to nouns.

Thus, for example, a set of four rules in pseudocode for identifying noun phrase is set forth in Table I below.

TABLE I

```
1   If the token is a member of Noun Phrase Tags
2       start to form a Noun Phrase.
3   If the token is a stop list noun or adjective
4       If the Noun-phrase length is 0
5           don't start the Noun Phrase
6       else
7           break the Noun Phrase.
8   If the token is a lowercase noun AND
9   the following token is an uppercase noun
10      break the Noun Phrase.
11  If the token is a member of Noun-phrase Tags
12      continue the Noun Phrase.
```

In Table I, lines 1–2 represent a first rule and provide for identifying as a "beginning of a noun phrase" those candidate tokens having a part-of-speech tag functionally related to noun word forms. That is, the first rule tags as the beginning of a noun phrase those tokens having a part-of-speech tag selected from the group of part-of-speech tags, including: singular common noun, adjective, ordinal number, cardinal number.

Lines 3–7, in Table I, represent a second rule. The second rule provides for identifying as an "end of the noun phrase" those candidate tokens having a part-of-speech tag selected from the group consisting of stoplist nouns and adjectives. The default implementation of the second rule contains the two stoplist nouns (i.e., one and ones) and one stoplist adjective (i.e., such). In particular applications, however, the user may introduce user-defined stoplist nouns and adjectives. For example, a user may chose to treat semantically vague generic nouns such as use and type as stoplist nouns.

In addition, lines 8–10 represent a third rule. This third rules specifies that module 53 of processor 47 is to identify as an "end of the noun phrase" those selected tokens having a part-of-speech tag of noun and having a Capcode Field identification of "000" (i.e., lowercase), when the selected token is followed by an extracted token having a part-of-speech tag of noun and having a Capcode Field identification of "001" (initial uppercase) or "010" (i.e., all uppercase). Thus, in general, the third rule demonstrates identifying the end of a noun phrase through analysis of a group of tokens surrounding a candidate token, and the third rule demonstrates identifying the end of a noun phrase through analysis of the part-of-speech tags and grammatical features of tokens in the window of sequential tokens.

The fourth rule, represented by lines 11–12 in Table I, provides for identifying as a "middle of the noun phrase" those selected tokens having a part-of-speech tag functionally related to noun word forms and following an extracted token identified as part of the noun phrase. For example, a token having a part-of-speech tag functionally related to noun word forms and following a token that has been identified as the beginning of the noun phrase is identified as a token contained within the middle of the noun phrase.

In operation, module 53 in conjunction with processor 47 applies each rule in Table I to each token extracted from the stream of natural language text. These rules allow module 53 to identify those tokens which are members of a noun phrase, and the relative position of each token in the noun phrase. The rules illustrated in Table I are not language-specific. However, other tables exist which contain language-specific rules for identifying noun phrases. Table II–VI, as set forth below, contain language-specific rules.

TABLE II

English Language Noun-Phrase Rules

```
1  If the token is uppercase AND
2  the token has a Part-of-speech Tag of Singular Adverbial Noun AND
3  the preceding token is a noun
4      break the Noun Phrase
5  If the token is an adjective AND
6  the preceding token is a non-possessive noun
7      break the Noun Phrase
8  If the token is "of" or "&" AND
9  the preceding token is an uppercase noun AND
10 the following token is an uppercase noun
11     form a Noun Phrase starting with the preceding token and
12     continue the Noun Phrase as long as Noun Phrase Tags are
13     encountered.
```

Table II contains a group of rules, in pseudocode, specific to the English language. For example, lines 1–4 specify a first rule for identifying the end of a noun phrase, lines 5–7 recite a second rule for identifying the end of a noun phrase, and lines 8–13 specify a third rule for identifying the beginning and for identifying the middle of a noun phrase.

TABLE III

German Language Noun-Phrase Rules

```
1  If the token is an adjective AND
2  the preceding token is a noun AND
3  the following token is a member of Noun Phrase Tags
4      break the Noun Phrase
```

Table III contains a group of rules, in pseudocode, specific to the German Language. For example, lines 1–4 specify a rule for identifying the end of a noun phrase.

TABLE IV

Italian Language Noun-Phrase Rules	
1	If the token is "di" AND
2	the preceding token is a noun AND
3	the following token is a lowercase noun
4	form a Noun Phrase starting with the preceding token and
5	continue the Noun Phrase as long as Noun Phrase Tags are
6	encountered.

Table IV contains a group of rules, in pseudocode, specific to the Italian Language, for example, lines 1–6 specify a rule for identifying the end of a noun phrase.

TABLE V

French and Spanish Noun Phrase Rules	
1	If the token is "de" AND
2	the preceding token is a noun AND
3	the following token is a lowercase noun
4	form a Noun Phrase starting with the preceding token and continue
5	Noun Phrase as long as Noun Phrase Tags are encountered.

Table V contains a group of rules, in pseudocode, specific to the French and Spanish Languages. For example, lines 1–5 recite a rule for identifying the beginning and the middle of a noun phrase.

TABLE VI

French and Spanish and Italian Noun-Phrase Rules	
1	If the token is an adjective AND
2	the preceding token is a noun AND
3	the following token is a noun
4	break the Noun Phrase

Table VI contains a group of rules, in pseudocode, specific to the French and Spanish and Italian languages. For example, lines 1–4 recite a rule for identifying the end of a noun phrase.

After action box **253** of FIG. **8**, control proceeds to decision box **254** of FIG. **8**. At decision box **254** the processor **47** identifies whether the user requested application of the agreement rules to the noun phrase identified in action box **253**. If the user did not request application of the agreement rules, control branches to decision box **256**. If the user did request application of the agreement rules, logical control proceeds to action box **255** wherein the agreement rules are applied.

At action box **255** the agreement checking module **57** of the processor **47** ensures that the tokens within the identified noun phrase are in agreement. Although English has no agreement rules, other languages such as German, French and Spanish require agreement between the words contained within a noun phrase. For example, French and Spanish require gender and number agreement within the noun phrase, while German requires gender, number, and case agreement within the noun phrase. The grammatical features concerning gender, number, and case agreement are supplied by the grammatical feature fields of the word data table.

FIG. **10** illustrates a pseudocode listing that processor **47** executes to ensure agreement between the various members contained within an identified noun phrase. In particular, processor **47** iteratively checks whether a first identified part of a noun phrase agrees with a second identified part of the noun phrase that immediately follows the first identified part in the stream of text. As described below, processor **47** ensures that each particular extracted token within the noun phrase agrees with all other extracted tokens contained in the noun phrase.

Pictorially, given a series of tokens with their associated agreement tags as shown below, where all tokens shown are valid candidates for being in the noun phrase, it would be possible to form a noun phrase that started with the token |i–2| and continued to the token |i+1| because they all agree with respect to the agreement tags of "Singular, Feminine".

	Agreement Tags	Agreement Tags	Agreement Tags
i–2	Plural, Masculine	Singular, Masculine	Singular, Feminine
i–1	Plural, Masculine	Singular, Feminine	Plural, Feminine
i	Singular, Feminine	Singular, Masculine	Plural, Masculine
i+1	Singular, Feminine		

In one embodiment for checking agreement, two temporary array areas, **temp1** and **temp2**, are proposed for storing the tokens while agreement is iteratively checked between the identified parts of the noun phrase.

The token [i–2], identified as the "beginning of the noun phrase" has all of its agreement tags copied to a temporary area, **temp1**.

temp1	Plural, Masculine	Singular, Masculine	Singular, Feminine
temp2			

All agreement tags for the next token, token [i–1], whose values agree with **temp1** area are placed in a second temporary area, **temp2**.

temp1	Plural, Masculine	Singular, Masculine	Singular, Feminine
temp2	Plural, Masculine	Singular, Feminine	

As long as there are some identified agreement tags in **temp1** and **temp2**, agreement has passed and the noun phrase can continue to be checked. If there is no match, agreement fails and the noun phrase is broken. When the noun phrase is broken, the last token that agrees with the previous tokens in the noun phrase is re-identified as the "end of the noun phrase".

In the current case being examined, there was agreement between **temp1** and **temp2**, so that the contents of **temp2** are copies of **temp1**, and the next token is retrieved.

temp1	Plural, Masculine	Singular, Feminine
temp2		

All agreement tags for the next token [i] whose values agree with **temp1** are placed in the second temporary area, **temp2**. When this is done, the temporary areas contain:

temp1	Plural, Masculine	Singular, Feminine
temp2	Singular, Feminine	Plural, Masculine

Because token [i–2], token [i–1], and token [i] all have the above listed agreement tags in common, the contents of the

temp2 area are copied to temp1, and the next token is retrieved.

temp1	Singular, Feminine	Plural, Masculine
temp2		

All agreement tags for the next token [i+1] whose values agree with temp1 are placed in a second temporary area, temp2. When this is done, the second temporary areas contain:

temp1	Singular, Feminine	Plural, Masculine
temp2	Singular, Feminine	

Because the token [i−2], token [i−1], token [i], and token [i+1] all have these agreement tags in common, the contents of the temp2 area are copied to temp1, and the next token is retrieved.

temp1	Singular, Feminine
temp2	

At this point, noun phrase processing ends in our example. All the tokens from token [i−2] to token [i+1] had at least one agreement tag in common, and thus passed the agreement test.

In a further embodiment, the agreement checker **57** of the processor **47** creates a "supertag" when checking agreement in accordance with action box **255** of FIG. **8**. The supertags allow the agreement module **57** to quickly identify whether the extracted tokens fail to agree, or whether they may agree. In particular, a supertag is created for each extracted word contained within the identified noun phrase by logically OR'ing together all the agreement tags associated with each identified token in the noun phrase.

A supertag associated with one token in the noun phrase is then compared against the supertag associated with the following token in the noun phrase to see if any form of agreement is possible. A form of agreement is possible if the required number, gender, and case parameters agree or contain potential agreements between each of the supertags. If the required number, gender, and case parameters contained in the supertags do not agree, then agreement is not possible. By making this comparison, it can be quickly determined whether or not agreement may exist between the tokens or whether agreement is impossible.

After action box **255**, logical flow proceeds to decision box **256**. At decision box **256** the processor **47** identifies whether the user requested application of the truncation rules to the noun phrase identified in action box **253**. If the user did not request application of the truncation rules, control branches to action box **258**. If the user did request application of the truncation rules, logical control proceeds to action box **257** wherein the truncation rules are applied.

At action box **257**, the truncator module **61** of the processor **47** truncates the identified noun phrases. In one aspect of the invention, as illustrated by the pseudocode listing of FIG. **11**, truncator **61** truncates noun phrases exceeding two words in length which satisfy a specific set of rules. In accordance with another aspect of the invention, the truncator **61** removes tokens within the noun phrase that fail to agree with the other tokens within the noun phrase. Preferably, this operation is achieved by the truncator module **61** operating in conjunction with the agreement checking module **57**. For example, agreement module **57** identifies those tokens within the noun phrase that are in agreement and those tokens that are not in agreement, and truncator module **61** re-examines which tokens belong in the noun phrase based upon the agreement analysis of agreement checking module **57**. Thus truncator module **61** truncates from the noun phrase the set of tokens following, and including, a token that does not agree with the preceding members of the identified noun phrase.

At action box **258**, processor **47** outputs the tokens extracted from the input stream of natural language text into the output buffer **19** of the application program interface **11**. Processor **47** also generates the token list **17** that correlates the input buffer of text **15** with the output buffer **19**, and places the token list **17** into the application program interface. The generated token list **17** comprises an array of tokens that describe parameters of the input and output data. The parameters associated with each token include the part-of-speech tags, the grammatical features, and the noun-phrase member tags. With this data, processor **30** in digital computer **12** is able to output to display **20** the identified noun phrases contained within the input stream of natural language text.

FIG. **12** illustrates an example of the operation of the noun-phrase analyzer **13** having an input buffer **400**, a token list **402**, an output buffer **404**, and identified noun phrases **406**. In particular, input buffer **400** contains a natural language text stream reading The cash flow is strong, the dividend yield is high, and. Token list **402** contains a list of tokens, wherein the tokens cash and dividend are identified as the "beginning of a noun phrase", and wherein the tokens flow and yield are identified as the "end of a noun phrase". Output buffer **404** contains a list of the lexical expressions found in the input buffer **400**, and box **406** contains the identified noun phrases cash flow and dividend yield.

FIG. **12** demonstrates the ability of the noun-phrase analyzer **10** to identify groups of words having a specific meaning when combined. Simply tokenizing the word in the stream of text and placing them in an index could result in many irrelevant retrievals.

FIG. **13** illustrates a pseudocode listing for implementing a morphological analyzer/generator **2**. In particular, the morphological analyzer can contain a processor **30** implementing the pseudocode listing of FIG. **13** as stored in memory **12**. Additional tables, as illustrated in FIG. **4A–4C**, necessary for the implementation of morphological analyzer/generator **2** can also be stored in memory element **12**.

Lines **1** and **54** of the pseudocode listing in FIG. **13** form a first FOR-LOOP that is operational until the noun form, the verb form, and the adverb/adjective form of the candidate word are each processed. In operation, processor **30** implements the conditions within the first FOR-LOOP of lines **1** and **54** by accessing the FIG. **3** representative entry **33** associated with the candidate word. The representative entry **33** includes a noun pattern field **46**, a verb pattern field **48**, and an adjective/adverb pattern field **50**. Each of the fields (e.g., **46**, **48**, and **50**) identifies a particular morphological transform in FIG. **4C**.

Lines **2–4** of the pseudocode listing contain steps for checking whether morphological paradigms associated with each particular grammatical field being processed (i.e. noun, verb, adjective/adverb) exist. The steps can be implemented by processor **30** accessing the FIG. **3** representative entry of the candidate word and identifying whether the fields **46**, **48**, **50** identify a valid morphological paradigm.

Lines 5–9 of the pseudocode of FIG. 13 include a logical IF-THEN-ELSE construct for determining the morphological paradigms associated with the candidate word. In particular, these steps form a variable called "LIST" that identifies the locations of paradigms. "LIST" can include one location in column 73 of FIG. 4C, or "LIST" can include a portmanteau rule identifying a plurality of locations in column 73.

Lines 10 and 53 of the pseudocode listing form a second FOR-LOOP nested within the first FOR-LOOP of lines 1 and 54. The second FOR-LOOP of lines 10 and 53 provide a logical construct for processing each of the paradigms contained in "LIST".

Lines 11 and 52 form a third nested FOR-LOOP that processes each candidate word once for each part-of-speech tag of the candidate word (identified as "POS tag" in FIG. 13). The part-of-speech tags of the candidate word (i.e. "POS tag") are identified by the POS Combination Index Field 44 of FIG. 3 that is associated with the candidate word.

In one aspect of the invention, lines 12–18 include steps for identifying morphological transforms of the candidate word given a part-of-speech tag for the candidate word and given a morphological paradigm for the candidate word. For example, the pseudocode instructions determine whether the baseform part-of-speech tag of the morphological transform (identified as "BASE POS" in FIG. 13) matches the part-of-speech tag of the candidate word. If a match is found, then the morphological transform is marked as a possible morphological transform for the candidate word, and the candidate word can be identified as a baseform.

Lines 27 and 51 of FIG. 13, in accordance with another aspect of the invention, contain a further nested FOR-LOOP. This FOR-LOOP operates upon each of the morphological transforms listed in the particular paradigm from 'LIST' that is currently being processed.

Further in accordance with the invention, each morphological transform within the current paradigm being processed is inspected to determine whether the morphological transform is an appropriate morphological transform for the candidate word. In particular, as illustrated by pseudocode lines 28–31, processor 30 identifies an appropriate morphological transform based upon whether a parameter of the candidate word matches a morphological pattern contained within a selected morphological transform

For instance, line 28 of the pseudocode determines whether the part-of-speech tag of the candidate word matches the part-of-speech tag of the morphological transform. If a match exists, the morphological transform is identified as an applicable transform for the candidate word.

In accordance with another embodiment of the invention, as shown in pseudocode lines 28–29 of FIG. 13, the processor 30 can identify an appropriate morphological transform based upon various parameter of the candidate word matching various morphological patterns contained within a selected morphological transform. The parameters of the candidate word can include: information contained within the representative entry 33, of FIG. 3; the length of the candidate word; and the identity of the character strings forming the candidate word, i.e. the suffixes, prefixes, and infixes in the candidate word. While the morphological patterns of a selected morphological transform are generally selected from the functional elements contained in the morphological transform. Thus, the morphological patterns can be selected from: a functional element defining the part-of-speech tag of the baseform; a functional element defining the character string to strip from a candidate word; a functional element defining the character string to add to a candidate word; and a functional element defining the part-of-speech tag of the morphologically transformed candidate word.

For example, the processor 30 can compare the suffix of a candidate word with the second functional element of the selected morphological transform, wherein the second functional element generally denotes the suffix to strip from the candidate word to form an intermediate baseform. In an alternative embodiment, the processor 30 can compare the prefix of the candidate word with the second functional element of the selected morphological transform. While in another embodiment the processor 30 compares the infix of the candidate word with the second functional element of the selected morphological transform. Following the comparison step, processor 30 then identifies those morphological transforms having morphological patterns matching the selected parameter of the candidate word as an appropriate transform for the candidate word.

Preferably, as illustrated in lines 28–31 of the FIG. 13 pseudocode listing the processor 30 only applies those transforms that both: (1) have a part-of-speech tag matching the part-of-speech tag of the candidate word; and (2) have a first character string to be removed from the candidate word that matches either a suffix, prefix, or infix in the candidate word.

According to a further embodiment of the invention, prefixation and infixation can be handled by separate structural elements in the system, as illustrated by pseudocode lines 32–35 of FIG. 13. Lines 32–35 illustrate a separate modular element for determining an applicable transform based on prefixation. Lines 32–35 first identifies whether the current morphological transform has the prefix flag set, as described in the discussion of FIB. 4C. If the prefix flag is set, a separate morphological prefix table containing morphological changes applicable to prefixes is referenced. The prefix table can be identified through the representative word entry 33 for the candidate word.

The prefix table will provide a list of baseform and inflection prefix pairs. To handle prefixation, the processor 30 will locate the longest matching prefix from one column in the prefix table, remove it, and replace it with the prefix from the other column. Preferably, these modifications will only be done when a morphological transform is tagged as requiring a prefix change. An analogous system can be created to address infixation.

Prefixation and infixation morphology are particularly applicable in Germanic languages, such as German and Dutch. In these languages the morphology of the word can change based upon the alteration of a character string in the beginning, middle, or end of the word. For example, German verbs display significant alternations in the middle and end of words: the verb einbringen (ein+bringen) forms its past participle as ein+ge+bracht, with the infixation (insertion) of the string ge between the verbal prefix and stem; and the transformation of the stem bringen into bracht.

The morphological analyzer/generator 2 illustrated in FIG. 13 provides a system capable of morphologically transforming words found within natural language text. For example, the multilingual text processor 10 of FIG. 1 can extract the candidate word drinks from a stream of text and forward the candidate word to analyzer/generator 2 through interface 11. The text processor 10 can further identify a representative entry 33 for the candidate word. Once a representative entry is located, the text processor 10 can provide information concerning the word drinks, such as the parts-of-speech and inflectional paradigms. In particular, the text processor 10 determines the parts-of-speech of drinks to

be noun plural and verb 3rd singular present; and the text processor determines the locations of a noun inflectional paradigm, a verb inflectional paradigm, an adjective/adverb paradigm, and a derivational paradigm.

After the text processor **10** obtains the data related to the candidate word drinks, the text processor can generate the appropriate morphological transforms in accordance with the pseudocode listing of FIG. **13**. The morphological analyzer/generator **2** first addresses the noun inflectional paradigm, and determines that the noun paradigm has only one paradigm. Analyzer/generator **2** then processes the candidate word by applying the inflectional transforms contained within the identified noun paradigm to each part-of-speech of the candidate word drinks. The inflectional transforms within the noun paradigm are applied by first determining which inflectional transforms should be applied, and by then applying those inflectional transforms to generate inflectional baseforms.

For instance, the candidate word contains a part-of-speech of noun plural which must first be matched with particular inflectional transforms contained within the noun paradigm. The matching can be accomplished, in one embodiment, by comparing the parts-of-speech associated with a particular transform to the part-of-speech of the candidate words. Thus, analyzer/generator **2** compares the current part-of-speech of the candidate word, i.e., noun plural, to the part-of-speech tags associated with the inflectional transforms stored in the noun inflectional paradigm. The analyzer determines: (1) the baseform part-of-speech of the noun paradigm is noun singular, that does not match the part-of-speech tag of the candidate word; (2) the first inflectional transform has as associated part-of-speech tag of noun singular possessive, that does not match the part-of-speech tag of the candidate word; and (3) the second inflectional transform has an associated part-of-speech tag of noun plural, that does match the associated part-of-speech tag of the candidate word. These comparison steps indicate that only the second inflectional transform matched the noun plural part-of-speech of the candidate word, and that therefore only the second inflectional transform contained within the noun paradigm is applied.

Analyzer/generator **2** then continues to process the candidate word by applying the inflectional transforms contained within the identified verb paradigm and the identified adjective/adverb paradigm. The verb paradigm contains one paradigm having a baseform and two inflectional transforms, while the candidate word is associated with a potentially matching part-of-speech tag of verb 3rd singular present. The baseform part-of-speech tag of the verb inflectional paradigm is "verb infinitive", that does not match the part-of-speech tag of the candidate word. The part-of-speech tag of the first inflectional transform is verb present participle, that does not match the part-of-speech tag of the candidate word. But, the part-of-speech tag of the second inflectional transform is verb 3rd singular present, that does match the part-of-speech tag of the candidate word. Thus, the inflectional transform contained within the second rule of the verb inflectional paradigm is applied to the candidate word.

After the application of the noun paradigm and the verb paradigm, the analyzer **2** processes the transforms contained within the adjective/adverb paradigm. In this particular case, the adjective/adverb paradigm is blank, thereby completing the inflectional transformation of the candidate word drinks.

FIG. **14** depicts a processing sequence for the uninflection module **5** for generating inflectional baseforms that begins at step **300**. At step **302** the candidate word for the inflectional analysis is obtained. Preferably, the candidate word is obtained from a stream of natural language text by tokenizer **43** as described in connection with FIG. **6**. After step **302**, logical flow proceeds to step **304**.

At step **304** the processor **30** obtains data relevant to the candidate word. This data is obtained by first finding a substantially equivalent expression to the candidate word in the word data table **31**. The substantially equivalent expression in the word data table **31** is then accessed to obtain an associated representative entry **33**. A representative entry **33** contains data such as the part-of-speech combination index, the noun inflection paradigms, the verb inflection paradigms, and the adjective/adverb inflection paradigms. The data obtained from representative entry **33** can also identify portmanteau paradigms that act as branching points to multiple numbers of other paradigms. At action box **310**, the flow chart indicates the beginning of the analysis of each paradigm.

At steps **312** and **314** the system determines whether the part-of-speech of the candidate word is in the same class as the current paradigm. For example, the processor determines whether the part-of-speech of the candidate word is the same as the part-of-speech of the paradigm identified by either the noun field **46**, the verb field **48**, or the adjective/adverb field **50** in the representative entry **33**. If the part-of-speech of the candidate word is not in the same class as the current paradigm, logical flow branches back to action block **312**. If the part-of-speech tag of the candidate word agrees with the current paradigm, then logical flow proceeds to decision box **316**.

Decision box **316** illustrates one preferred embodiment of the invention, wherein the candidate word is compared to the paradigm's baseform. If the candidate word matches the paradigm baseform, logical flow proceeds to decision box **328**. That is, if the candidate word matches the subparadigm's baseform no uninflection is necessary. In many situations, however, the candidate word will not match the paradigm baseform. When the candidate word differs from the paradigm baseform, logical flow proceeds to action box **318**.

Action box **318** begins another logical FOR-LOOP wherein each inflectional transform is processed. In accordance with FIG. **14**, logical flow proceeds from box **318** to decision box **320**.

At decision box **320** two aspects of the invention and a preferred embodiment are illustrated. In particular, action box **320** indicates that the part-of-speech tag of the candidate word can be compared with the fourth functional element of the inflectional transform (i.e. the functional element specifying the part-of-speech of the transform). If the part-of-speech tags matches, then logical flow proceeds to action box **322**. However, if the part-of-speech tags differ, logical flow branches back to box **18**. According to a further aspect of the invention, as illustrated in action box **320**, the ending character strings of the candidate word and the second functional element of the inflectional transform (i.e. the functional element specifying the suffix to strip from the candidate word) are compared. If the character strings do not match, logical flow proceeds back to action box **318** while if the character strings match, logical flow proceeds to action box **322**. Preferably, as illustrated in FIG. **14**, the uninflectional module **5** compares the part-of-speech tags associated with the inflectional transform and the candidate word, and the uninflectional module **5** compares the character strings associated with the inflectional transform and the candidate word. According to this preferred embodiment, only if the part-of-speech tags match and the character strings match does logical flow proceed to action box **322**.

At step 322, uninflection module 5 implements a strip and add algorithm to form the inflectional baseform of the candidate word. The strip and add algorithm is obtained from the inflectional transform currently being processed. The transform currently being processed indicates a particular character string to be removed from the candidate word and a subsequent character string to be added to the character word to form the inflectional baseform. After step 322, logical flow proceeds to decision box 324.

Decision box 324 is an optional step involving prefixation. If prefixation operations are requested by the user, boxes 324 and 326 will be activated. At decision box 324 the processor 30 identifies whether the inflectional transform currently being considered has a prefixation rule associated with it. If the transform does contain the prefixation rule logical flow proceeds to action box 326, otherwise logical flow proceeds to action box 328. At action box 326 the prefix is removed from the baseform in accordance with the inflectional transform. Logical flow then proceeds to box 328.

Steps 328, 330, 332, and 334 are optional steps demonstrating one implementation of the coupling between the inflection module 4, the uninflectional module 5, the derivation expansion module 6, and underivation (derivation reduction) module 7.

In particular, action box 328 identifies whether the user has requested underivation (derivation reduction). If underivation (derivation reduction) has been requested, logical flow proceeds to action box 330, otherwise flow proceeds to decision box 332. At action box 330 the candidate word undergoes underivation (derivation reduction) in accordance with the flowchart identified in FIG. 16. Following underivation (derivation reduction), logical flow proceeds to decision box 332. At decision box 332 the processor identifies whether inflection has been requested. If inflection was requested, logical flow proceeds to action box 334, wherein the candidate word undergoes inflection analysis in accordance with the steps illustrated in FIG. 15. If inflection was not requested, logical flow proceeds directly to action box 336.

At action box 336 the logical FOR-LOOP for the inflectional transform ends and at action box 338 the logical FOR-LOOP for the paradigms ends, thereby completing the uninflection routine.

FIG. 15 depicts a processing sequence for the inflection module 4 of the morphological analyzer of FIG. 1. The inflection analysis begins at step 340 and logical control proceeds to action box 342. At action box 342 the inflection module 4 obtains an inflectional baseform of a candidate word. The inflectional baseform can be obtained, for example, from a candidate word which is processed by the uninflection module 5 in accordance with FIG. 14. After action box 342, logical flow proceeds to action box 344.

Box 344 begins a logical FOR-LOOP that is applied to each inflectional transform in the paradigm associated with the candidate word.

At action box 346 and 348 the inflection module attends to prefixing if prefixing processing was requested by the user of the text processing system 10. Decision box 346 determines whether a prefixing rule is contained within the inflectional transform, and if such a prefixing rule is present the rule is applied at action box 348. After boxes 346 and 348, logical flow proceeds to box 350.

At step 350 characters are removed from the baseform to form an intermediate baseform, and at step 352 characters are added to the intermediate baseform to form the inflected pattern. Thereafter, action box 354 assigns the part-of-speech tag associated with the applied inflectional transform to the newly generated inflected form. Action box 356 ends the FOR-LOOP begun at action box 344.

FIG. 16 depicts a further processing sequence for the underivation (derivation reduction) module 7 of the morphological analyzer 2, that begins at step 360. At action box 362 underivation (derivation reduction) module 6 obtains a baseform of the candidate word. The baseform can be obtained from the uninflection module 5. After action box 362, control proceeds to box 364.

Decision box 364 identifies whether the derivation paradigm is an empty set or whether it contains morphological transforms. In particular, if derivational paradigms do not exist for this baseform, logical flow proceeds to action box 396 ending the underivation (derivation reduction) process. However, if the derivation paradigm is not blank, logical control continues to box 366.

Box 366 begins a logical FOR-LOOP for processing each derivational paradigm. After box 366, control proceeds to decision box 368.

Decision box 368 examines whether the candidate word is a derivational route or not. Determination of the derivation route characteristics of the word can be performed by analyzing the information contained within the representative entry 33 associated with the candidate word. For example, the Is Derivation Field 38 of FIG. 3 identifies whether the candidate word is a derivational route. If the candidate word is marked as a derivational route, logical flow proceeds to action box 394, otherwise logical flow proceeds to action box 376.

Action box 376 begins a logical FOR-LOOP for processing each derivational transform in the subparadigm. After action box 376, logical flow proceeds to decision box 378.

Decision box 378 determines whether the derivational transform includes a character string matching the candidate word's ending string. If no match is found, logical flow will proceed to action box 376, otherwise logical flow will proceed onto box 380.

At action box 380, the derivational reduction module 7 implements the transform for changing the candidate word into the derivational baseform of the word. This process is implemented by removing a first character string from the candidate word and adding a second character string to the candidate word in accordance with the derivational transform. At box 382, the newly transformed word is marked as a derivational root. After box 382, flow proceeds to decision box 384.

Boxes 384 and 386 are optional boxes providing prefixing adjustments to the newly formed derivational root. For example, decision box 384 determines whether a prefixing rule exists within the derivational transform and if such a prefixing rule exists then insures that logical flow proceeds to action box 386. At action box 386, the prefix is removed to generate a more accurate derivational root. After the implementation of optional boxes 384 and 386, logical flow proceeds on to box 392.

At box 392, the FOR-LOOP which began with box 376 ends. Box 394 ends the logical FOR-LOOP associated with action box 366. Once each of the paradigms has been completely processed logical flow will proceed from box 394 to box 396. Box 396 indicates the end of the underivation (derivation reduction) module.

FIG. 17 illustrates a processing sequence of derivation expansion module 6 for generating derivatives of the candidate word. Operation of the derivation expansion module begins at step 400, after which logical control proceeds to action box 402. At action box 402 the derivation expansion

module obtains the derivational root of the candidate word. This root can be obtained from the underivation (derivation reduction) module 7 of FIG. 16.

After action box 402, control proceeds to action box 404. Box 404 provides a logical FOR-LOOP for each derivational transform in the paradigm associated with the derivational root obtained at action box 402. After action box 404, control proceeds to decision box 406.

Boxes 406 and 408 illustrate optional prefixing control boxes. These control boxes are implemented if the user requests prefixing. Following action box 408 control proceeds to action box 410.

At action box 410, derivation expansion module 6 removes characters from the derivational root in accordance with the derivational transform associated with the paradigm currently being processed. After box 410, logical control passes to action box 412. At action box 412, a string of characters is added to the intermediate root formed in action box 410 in accordance with the current derivational transform. After box 412 control proceeds to box 414. At action box 414 a part-of-speech tag is assigned to the newly generated derivational expansion in accordance with the derivational transform. Following box 414, control proceeds to action box 420. Action box 420 ends the FOR-LOOP associated with action box 404, thereby ending the derivation expansion processing.

While the invention has been shown and described having reference to specific preferred embodiments, those skilled in the art will understand that variations in form and detail may be made without departing from the spirit and scope of the invention.

Having described the invention, what is claimed as new and secured by Letters Patent is:

1. A programmed data processing method for generating a morphologically related word from a candidate word using a first addressable table containing a list of lexical expressions and a second addressable table containing a list of paradigms, each paradigm having at least one transform that includes at least a first morphological pattern and a second morphological pattern, said method comprising the steps of:

locating in the first addressable table a first lexical expression substantially equivalent to the candidate word,

identifying a first paradigm in the second addressable table as a function of the located first lexical expression,

matching a transform in the identified first paradigm with the candidate word,

forming an intermediate baseform by stripping a first character string from the candidate word, the first character string being defined by the first morphological pattern included with the matched transform, and

generating a morphological baseform of the candidate word by adding a second character string to the formed intermediate baseform, the second character string being defined by the second morphological pattern included with the matched transform.

2. A method in accordance with claim 1, wherein said matching step further comprises:

identifying a parameter of the candidate word,

selecting a morphological pattern for each transform, and

matching a transform with the candidate word when the identified parameter of the candidate word matches the selected morphological pattern.

3. A method in accordance with claim 2, wherein the identified parameters of the candidate word are selected from the group consisting of: part-speech-tags, grammatical features, the length of the candidate word, suffixes, prefixes, and infixes.

4. A method in accordance with claim 2, wherein the second addressable table includes part-of-speech tags associated with each transform and said matching step further comprises:

identifying the parameter of the candidate word as a part-of-speech tag of the candidate word,

selecting a part-of-speech tag as the morphological pattern for each transform and

matching a transform having an associated first part-of-speech tag with the candidate word when the first part-of-speech tag matches the identified part-of-speech tag of the candidate word.

5. A method in accordance with claim 1, wherein the intermediate baseform formed varies as a function of the matched transform.

6. A method in accordance with claim 1, further comprising the steps of:

locating a portmanteau paradigm in the second addressable table as a function of the first lexical expression, the portmanteau paradigm including the locations of a plurality of paradigms, and

identifying at least a first paradigm selected from the plurality of paradigms included in the portmanteau paradigm.

7. A method in accordance with claim 6, wherein said portmanteau paradigm includes the location of a noun paradigm, a verb paradigm, and an adjective/adverb paradigm.

8. A method in accordance with claim 7, wherein the second addressable table includes a baseform part-of-speech tag associated with each morphological transform and wherein said matching step further includes matching the candidate word only with those morphological transforms in the identified noun paradigm having an associated baseform part-of-speech tag of noun.

9. A method in accordance with claim 7, wherein the second addressable table includes a baseform part-of-speech tag associated with each morphological transform and wherein said matching step further includes matching the candidate word only with those morphological transforms in the identified verb paradigm having an associated baseform part-of-speech tag of verb.

10. A method in accordance with claim 7, wherein the second addressable table includes a baseform part-of-speech tag associated with each morphological transform and wherein said matching step further includes matching the candidate word only with those morphological transforms in the identified adjective/adverb paradigm having an associated baseform part-of-speech tag selected from the group consisting of adverb and adjective.

11. A method in accordance with claim 1, wherein said step of forming an intermediate baseform further comprises stripping a suffix character string from the end of the candidate word.

12. A method in accordance with claim 1, wherein said step of forming an intermediate baseform further comprises stripping a prefix character string from the front of the candidate word.

13. A method in accordance with claim 1, wherein said step of forming an intermediate baseform further comprises stripping an infix character string from the middle of the candidate word.

14. A method in accordance with claim 13 wherein applying the list of transforms further comprises:

forming a second intermediate baseform by stripping a first inflectional character string from the generated

inflectional baseform in accordance with a selected transform, and

generating an inflected form of the generated inflectional baseform by adding a second inflectional character string to the second intermediate baseform.

15. A method in accordance with claim 1, further comprising the step of extracting the first lexical expression from a stream of natural language text prior to said locating step.

16. A programmed data processing method for generating an inflectional baseform from a candidate word using a first addressable table containing a list of lexical expressions and a second addressable table containing a list of paradigms, each paradigm having at least one inflectional transform that includes at least a first inflectional pattern and a second inflectional pattern, said method comprising the steps of:

locating in the first addressable table a first lexical expression substantially equivalent to the candidate word,

identifying a first paradigm in the second addressable table as a function of the located first lexical expression,

matching an inflectional transform in the identified first paradigm with the candidate word,

forming an intermediate baseform by stripping a first character string from the candidate word, the first character string being defined by the first inflectional pattern included with the matched transform, and

generating an inflectional baseform of the candidate word by adding a second character string to the formed intermediate baseform, the second character string being defined by the second inflectional pattern included with the matched inflectional transform.

17. A method in accordance with claim 16, further comprising the step of creating the inflected forms of the generated inflected baseform by applying the list of transforms in the identified first paradigm to the generated inflectional baseform.

18. A programmed data processing method for generating a derivationally related word from a candidate word using a first addressable table containing a list of derivational paradigms, each derivational paradigm having at least one derivational transform and an associated derivational pattern, said method comprising the steps of:

identifying a first derivational paradigm in the first addressable table as a function of the candidate word,

matching a derivational pattern in the identified first derivational paradigm with the candidate word,

forming an intermediate derivational baseform by stripping a first character string from the candidate word in accordance with a first derivational transform associated with the matched derivational pattern, and

generating the derivational baseform of the candidate word by adding a second character string to the formed intermediate derivational baseform in accordance with the first derivational transform.

19. A method in accordance with claim 18, wherein said derivational pattern matching step further comprises identifying a derivational pattern that matches a suffix pattern in the candidate word.

20. A method in accordance with claim 19, wherein said step of forming an intermediate baseform further comprises stripping a suffix character string from the end of the candidate word.

21. A method in accordance with claim 18, wherein said derivational pattern matching step further comprises identifying a derivational pattern that matches a prefix pattern in the candidate word.

22. A method in accordance with claim 21, wherein said step of forming an intermediate baseform further comprises stripping a prefix character from the front of the candidate word.

23. A method in accordance with claim 18, wherein said derivational pattern matching step further comprises identifying a derivational pattern that matches an infix pattern in the candidate word.

24. A method in accordance with claim 23, wherein said step of forming an intermediate baseform further comprises stripping an infix character from the middle of the candidate word.

25. A method in accordance with claim 18 wherein said intermediate derivational baseform varies as a function of the derivational patterns contained within an identified derivational paradigm.

26. A method in accordance with claim 25 wherein applying the list of derivational transforms further comprises:

forming a second intermediate derivational baseform by stripping a first derivational character string from the generated derivational baseform in accordance with a selected derivational transform, and

generating a derivational expansion of the generated baseform by adding a second derivational character string to the second intermediate derivational baseform.

27. A method in accordance with claim 18 wherein said identifying step further comprises:

locating a first lexical expression substantially equivalent to the candidate word in a second table containing a list of lexical expressions, and

identifying the first derivational paradigm in the first addressable table as a function of the located first lexical expression.

28. A method in accordance with claim 27 wherein the first addressable table includes derivational root tags associated with each lexical expression, said locating step further comprising the step of tagging a candidate word when the first located lexical expression is a derivational root.

29. A method in accordance with claim 18, wherein said identifying step further comprises:

locating a first lexical expression substantially equivalent to the candidate word in a second table containing a list of lexical expressions,

determining the inflectional baseform of the first lexical expression, and

identifying the first derivational paradigm in the first addressable table as a function of the determined inflectional baseform.

30. A method in accordance with claim 18, further comprising the step of creating the derivational expansions of the generated derivational baseform by applying the list of derivational transforms in the identified first derivational paradigm to the generated derivational baseform.

31. An apparatus for generating morphologically related forms of a candidate word, said apparatus comprising:

A) a digital memory element, including

a first addressable table having a list of lexical expressions,

a second addressable table having a list of paradigms, each paradigm having at least one morphological transform that includes at least a first morphological pattern and a second morphological pattern, and

wherein each lexical expression listed in said first addressable table is associated with at least one paradigm listed in said second table,

B) a digital data processing element coupled with said digital memory element, said digital data processor including

first processing means for identifying a first paradigm for the candidate word by locating a lexical expression representative of the candidate word in said first addressable table,

second processing means for matching a first morphological transform in the first paradigm with the candidate word,

third processing means for forming an intermediate baseform by stripping a first character string from the candidate word, the first character string being defined by the first morphological pattern included with the matched transform, and

fourth processing means for generating a morphological baseform of the candidate word by adding a second character string to the formed intermediate baseform, the second character string being defined by the second morphological pattern included with the matched transform.

32. An apparatus according to claim 31, wherein said second processing means further comprises:

identifying means for identifying a parameter of the candidate word,

selecting means for selecting a morphological pattern for each transform, and

comparing means for comparing the parameter of the candidate word with the selected morphological pattern such that a transform is matched with the candidate word.

33. An apparatus according to claim 32, wherein the identifying means selects the parameter of the candidate word from the group of parameters consisting of: part-of-speech tags, grammatical features, length of the candidate word, suffixes, prefixes, and infixes.

34. An apparatus according to claim 32 wherein the second addressable table includes part-of-speech tags associated with each transform and the selected morphological pattern is the part-of-speech tag associated with each transform, the second processing means further comprising:

means for identifying a part-of-speech tag of the candidate word, and

means for comparing the part-of-speech tag of the candidate word with the part-of-speech tag associated with each transform, such that the candidate word is matched with a transform having an equivalent part-of-speech tag.

35. An apparatus according to claim 31, wherein the third processing means forms an intermediate baseform that varies as a function of the first morphological transform matching the candidate word.

36. An apparatus according to claim 31, further comprising:

portmanteau paradigm means stored in the second addressable table for identifying the location of a plurality of paradigms, and

fifth processing means for identifying a paradigm selected from the plurality of paradigms included in the portmanteau paradigm.

37. An apparatus according to claim 31, further comprising tokenizing means for extracting the candidate word from a stream of natural language text.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 5,794,177

DATED : August 11, 1998

INVENTOR(S) : Carus et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On the title page change Assignee to: Lernout & Hauspie Speech Products N.V.

Signed and Sealed this

Thirty-first Day of August, 1999

Attest:

Attesting Officer

Q. TODD DICKINSON

Acting Commissioner of Patents and Trademarks